**Name : ABITHA M**

**Batch : 2022 – 7607**

**Enrolment number : EBEON0722634453**

**Java**

Java is one of the most popular and commonly used programming languages. It is widely recognized for its performance, platform independence, and security

Java is used as a server-side language for back-end development, android development, desktop computing, games, and numerical computing. Various features of Java contribute to its wide use and popularity.

**Java Development kit (JDK)**

JDK is a software development environment used for making applets and Java applications. The full form of JDK is Java Development Kit. Java developers can use it on Windows, macOS, Solaris, and Linux. JDK helps them to code and run Java programs. It is possible to install more than one JDK version on the same computer.

**Java Virtual Machine (JVM):**

Java Virtual Machine (JVM) is an engine that provides a runtime environment to drive the Java Code or applications. It converts Java bytecode into machine language. JVM is a part of the Java Run Environment (JRE). In other programming languages, the compiler produces machine code for a particular system. However, the Java compiler produces code for a Virtual Machine known as Java Virtual Machine.

[**Features of Java Programming Language**](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#fea)

* [Inspired by C and C++](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#ins)
* [Simple and familiar](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#sim)
* [Object-oriented](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#oo)
* [Platform independent](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#pi)
* [Compiled and interpreted](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#ci)
* [Multi-threaded](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#mt)
* [Dynamic](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#dy)
* [Robust](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#rob)
* [Secure](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#sec)
* [High Performance](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#hp)
* [Portable](https://www-naukri-com.cdn.ampproject.org/v/s/www.naukri.com/learning/articles/features-of-java-programming-language/amp?usqp=mq331AQKKAFQArABIIACAw%3D%3D&amp_js_v=a9&amp_gsa=1#por)

**Object-Oriented**

Java is an object-oriented programming language. Object-oriented programming (OOP) is a programming paradigm that organizes software design around data, rather than functions and logic. An object is a data field with its own set of properties and behavior. For example, a student class can have variables like Name, Class, Roll Number, Father’s Name, Marks, and methods to store and retrieve these values.

**The basic concepts of OOPs are:**

* Object
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

### **Object**

An [object](https://www.javatpoint.com/object-and-class-in-java#object) is a real-world entity that has attributes, behavior, and properties. It is referred to as an instance of the class. It contains member functions, variables that we have defined in the class. It occupies space in the memory. Different objects have different states or attributes, and behaviors.

**Class**

Collection of objects is called class. It is a logical entity.A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

**Inheritance**

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

**Polymorphism**

If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

**Abstraction**

Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

In Java, we use abstract class and interface to achieve abstraction.

**Encapsulation**

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class

**Constructor**

The constructor and method both are different from each other. However, the constructor is used for initializing the object's state. Constructors can also contain data members and member functions in the same way as the method contains. The data members and member functions of the constructor are executed at the time of creating an object of that class.

The new keyword plays an important role in creating an instance of the class.

**Method**

The method is a block of code that is used for performing a particular task. It reduces the length and the repetitive code. It may or may not return a value to the caller.

By creating a method, we can reuse the code without retyping the code.

**Collections in Java:**

The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes ([ArrayList](https://www.javatpoint.com/java-arraylist), Vector, [LinkedList](https://www.javatpoint.com/java-linkedlist), [PriorityQueue](https://www.javatpoint.com/java-priorityqueue), HashSet, LinkedHashSet, TreeSet).

**List Interface**

List interface is the child interface of Collection interface. It inhibits a list type data structure in which we can store the ordered collection of objects. It can have duplicate values.

List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

**ArrayList**

The ArrayList class implements the List interface. It uses a dynamic array to store the duplicate element of different data types. The ArrayList class maintains the insertion order and is non-synchronized. The elements stored in the ArrayList class can be randomly accessed.

**LinkedList**

LinkedList implements the Collection interface. It uses a doubly linked list internally to store the elements. It can store the duplicate elements. It maintains the insertion order and is not synchronized. In LinkedList, the manipulation is fast because no shifting is required.

**Vector**

Vector uses a dynamic array to store the data elements. It is similar to ArrayList. However, It is synchronized and contains many methods that are not the part of Collection framework.

**Stack**

The stack is the subclass of Vector. It implements the last-in-first-out data structure, i.e., Stack. The stack contains all of the methods of Vector class and also provides its methods like boolean push(), boolean peek(), boolean push(object o), which defines its properties.

**Queue Interface**

Queue interface maintains the first-in-first-out order. It can be defined as an ordered list that is used to hold the elements which are about to be processed. There are various classes like PriorityQueue, Deque, and ArrayDeque which implements the Queue interface.

**PriorityQueue**

The PriorityQueue class implements the Queue interface. It holds the elements or objects which are to be processed by their priorities. PriorityQueue doesn't allow null values to be stored in the queue.

**Deque Interface**

Deque interface extends the Queue interface. In Deque, we can remove and add the elements from both the side. Deque stands for a double-ended queue which enables us to perform the operations at both the ends.

**ArrayDeque**

ArrayDeque class implements the Deque interface. It facilitates us to use the Deque. Unlike queue, we can add or delete the elements from both the ends.

ArrayDeque is faster than ArrayList and Stack and has no capacity restrictions.

**Set Interface**

Set Interface in Java is present in java.util package. It extends the Collection interface. It represents the unordered set of elements which doesn't allow us to store the duplicate items. We can store at most one null value in Set. Set is implemented by HashSet, LinkedHashSet, and TreeSet.

**HashSet**

HashSet class implements Set Interface. It represents the collection that uses a hash table for storage. Hashing is used to store the elements in the HashSet.

LinkedHashSet

LinkedHashSet class represents the LinkedList implementation of Set Interface. It extends the HashSet class and implements Set interface. Like HashSet, It also contains unique elements. It maintains the insertion order and permits null elements.

**SortedSet Interface**

SortedSet is the alternate of Set interface that provides a total ordering on its elements. The elements of the SortedSet are arranged in the increasing (ascending) order. The SortedSet provides the additional methods that inhibit the natural ordering of the elements.

**TreeSet**

Java TreeSet class implements the Set interface that uses a tree for storage. Like HashSet, TreeSet also contains unique elements. However, the access and retrieval time of TreeSet is quite fast. The elements in TreeSet stored in ascending order

**Java Code -Password Generator :**

package Generate;

import java.util.Scanner;

public class Generator {

boolean IncludeUpper=false;

boolean IncludeLower=false;

boolean IncludeNum=false;

boolean IncludeSym=false;

Generator alphabet;

public Generator(boolean IncludeUpper,boolean IncludeLower, boolean IncludeNum, boolean IncludeSym) {

alphabet=new Generator(IncludeUpper, IncludeLower, IncludeNum, IncludeSym);

}

public Password GeneratePassword(int length) {

final StringBuilder pass = new StringBuilder("");

final int alphabetLength = alphabet.getAbstract().length();

int max = alphabetLength - 1;

int min = 0;

int range = max - min + 1;

for(int i=0;i<length;i++) {

int index=(int)(Math.random() \* range) + min;

pass.append(alphabet.getAlphabet().charAt(index));

}

return new Password(pass.toString());

public static void printUsefulInfo() {

System.out.println();

System.out.println("Use a minimum password length of 8 or more characters if permitted");

System.out.println("Include lowercase and uppercase alphabetic characters, numbers and symbols if permitted");

System.out.println("Generate passwords randomly where feasible");

System.out.println("Avoid using the same password twice (e.g., across multiple user accounts and/or software systems)");

System.out.println("Avoid character repetition, keyboard patterns, dictionary words, letter or number sequences,\nusernames, relative or pet names, romantic links (current or past) and biographical information (e.g., ID numbers, ancestors' names or dates).");

System.out.println("Avoid using information that the user's colleagues and/or acquaintances might know to be associated with the user");

System.out.println("Do not use passwords which consist wholly of any simple combination of the aforementioned weak components");

}

public static void requestPassword() {

boolean IncludeUpper=false;

boolean IncludeLower=false;

boolean IncludeNum=false;

boolean IncludeSym=false;

String Input;

final Scanner in = new Scanner(System.in);

System.out.println();

System.out.println("Hello, welcome to the Password Generator :) answer"

+" the following questions by Yes or No \n");

while(true) {

System.out.println("Do you want Lowercase letters \"abcd...\" to be used? ");

Input = in.nextLine();

if (Input.equals("YES") || Input.equals("Yes") || Input.equals("yes")) {

IncludeLower=true;

}

else {

if (!(Input.equals("NO")) && (!Input.equals("No")) && (!Input.equals("no"))) {

PasswordRequestError();

break;

}

}

System.out.println("Do you want Uppercase letters \"ABCD...\" to be used? ");

Input = in.nextLine();

if (Input.equals("YES") || Input.equals("Yes") || Input.equals("yes")) {

IncludeUpper=true;

}

else {

if (!(Input.equals("NO")) && (!Input.equals("No")) && (!Input.equals("no"))) {

PasswordRequestError();

break;

}

}

System.out.println("Do you want Numbers \"1234...\" to be used? ");

Input = in.nextLine();

if (Input.equals("YES") || Input.equals("Yes") || Input.equals("yes")) {

IncludeNum=true;

}

else {

if ((Input.equals("NO")==false) && (Input.equals("No")==false) && (Input.equals("no")==false)) {

PasswordRequestError();

break;

}

}

System.out.println("Do you want Symbols \"!@#$...\" to be used? ");

Input = in.nextLine();

if (Input.equals("YES") || Input.equals("Yes") || Input.equals("yes")) {

IncludeSym=true;

}

else {

if ((Input.equals("NO")==false) && (Input.equals("No")==false) && (Input.equals("no")==false)) {

PasswordRequestError();

break;

}

}

//No Pool Selected

if(!IncludeUpper && !IncludeLower && !IncludeNum && !IncludeSym) {

System.out.println("You have selected no characters to generate your password at least one of your answers should be Yes");

break;

}

System.out.println("Great! Now enter the length of the password");

int length=in.nextInt();

final Generator generator = new Generator (IncludeUpper, IncludeLower, IncludeNum, IncludeSym);

final Password UserPass = generator.GeneratePassword(length);

System.out.println(UserPass);

in.close();

break;

}

}

public static void PasswordRequestError() {

System.out.println("You have entered something incorrect let's go over it again \n");

}

public static void checkPassword() {

String input;

final Scanner in = new Scanner(System.in);

System.out.print("\nEnter your password:");

input = in.nextLine();

final Password p = new Password(input);

System.out.println(p.calculateScore());

in.close();

}

public static void printMenu() {

System.out.println();

System.out.println("Enter 1 - Password Generator");

System.out.println("Enter 2 - Password Stength Check");

System.out.println("Enter 3 - Useful Information");

System.out.println("Enter 4 - Quit");

System.out.print("Choice:");

}

public static void main(String[] args) {

String Input;

final Scanner in = new Scanner(System.in);

System.out.println("Welcome to Ziz Password Services :)" );

printMenu();

while (in.hasNextLine()) {

Input = in.nextLine();

if (Input.equals("1")) {

requestPassword();

printMenu();

}

else if (Input.equals("2")) {

checkPassword();

printMenu();

}

else if (Input.equals("3")) {

printUsefulInfo();

printMenu();

}

else if (Input.equals("4")) {

break;

}

else {

System.out.println();

System.out.println("Kindly select one of the available commands" );

printMenu();

}

}

in.close();

}

}