**Types of Hacker:**

white Hat:

People who specialized hacking check the faults of the system

Grey hat:

Exploit a security to the attention of the owners

Black Hat:

People who break into the networks and harm the network and property

**Common Types of attacks:**

1. Malware:

Software programs designed to damage or do unwanted actions on a computer.

Common examples include: viruses, worms, trojan horses, spyware, and ransomware.

2. Phishing:

Attacks sent via email and ask users to click on a link and enter their personal data.

They include a link that directs the user to a dummy site that will steal a user’s information.

3. Password Attacks

Involves a third party trying to gain access to your systems by solving a user’s password.

4. Denial of Service Attacks:

Attackers send high volumes of data or traffic through the network until the network becomes overloaded and can no longer function.

5. Man in the Middle (MITM):

Information is obtained from the end user and the entity the user is communicating with by impersonating the endpoints in an online

information exchange (i.e. connection from smartphone to website).

6. Drive-by Downloads :

A program is downloaded to a user’s system just by visiting the site. It doesn’t require any type of action by the user to download.

**Collections in Java**

Java Collection Framework

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, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

An interface in Java is a blueprint of a class. It has static constants and abstract methods.

**Interfaces in java**

The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body.

It is used to achieve abstraction and multiple inheritance in Java.

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body

***Why use Java interface?***

There are mainly three reasons to use interface. They are given below.

It is used to achieve abstraction.

By interface, we can support the functionality of multiple inheritance.

It can be used to achieve loose coupling.

Java Lambda Expressions

Lambda expression is a new and important feature of Java which was included in Java SE 8.

It provides a clear and concise way to represent one method interface using an expression.

It is very useful in collection library. It helps to iterate, filter and extract data from collection.

interface Sayable{

public String say();

}

public class LambdaExpressionExample3{

public static void main(String[] args) {

Sayable s=()->{

return "I have nothing to say.";

};

System.out.println(s.say());

}

}

**#pragma Directive in C/C++**

**#pragma startup and #pragma exit**: These directives helps us to specify the functions that are needed to run before program startup( before the control passes to main()) and just before program exit (just before the control returns from main()).

|  |
| --- |
| #include<stdio.h>  void func1();  void func2();  #pragma startup func1  #pragma exit func2    void func1()  {     printf("Inside func1()\n");  }  void func2()  {      printf("Inside func2()\n");  }  int main()  {      printf("Inside main()\n");      return 0;  } |