**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **16/06/2020** | **Name:** | **Navya R** |
| **Course:** | **Cyber security** | **USN:** | **4AL16EC041** |
| **Topic:** | **What is cyber security?**  **Security and threats** | **Semester & Section:** | **8 A** |
| **Github Repository:** | **Navya-R** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  Cyber-attacks are becoming more sophisticated all the time, with cyber hackers coming up with new and determined methods of threat that are increasingly difficult to detect, making attacks more dangerous than ever before.  Statistics show that cybercrime is on the rise around the world. It’s estimated that by 2021 the annual cost of damages from[cybercrime will cost the world $6 trillion](https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/). That’s a significant jump from $3 trillion in 2015, with cyber-attacks now one of the most serious threats to any business.  No matter the size of your organization, whether you’re a start-up business or have scaled to a million-dollar company, you need to be aware of the risk of a cyber-attack.  What’s the motivation behind all this cybercrime? The results of studies done on cyberhacking show that the motivation behind 90% of attacks is about financial gain and espionage. Here’s a closer look at the most-breached industries, who is doing the hacking, and what type of data is being hacked. The Three Security Goals Are Confidentiality, Integrity, and Availability All information security measures try to address at least one of three goals:   * Protect the confidentiality of data * Preserve the integrity of data * Promote the availability of data for authorized use   These goals form the confidentiality, integrity, availability (CIA) triad, the basis of all security programs (see [Figure 2.1](javascript:popUp('/content/images/chap2_9780789753250/elementLinks/02fig01_alt.jpg'))). Information security professionals who create policies and procedures (often referred to as governance models) must consider each goal when creating a plan to protect a computer system. Integrity Models Integrity models keep data pure and trustworthy by protecting system data from intentional or accidental changes. Integrity models have three goals:   * Prevent unauthorized users from making modifications to data or programs * Prevent authorized users from making improper or unauthorized modifications * Maintain internal and external consistency of data and programs   An example of integrity checks is balancing a batch of transactions to make sure that all the information is present and accurately accounted for. Availability Models Availability models keep data and resources available for authorized use, especially during emergencies or disasters. Information security professionals usually address three common challenges to availability:   * Denial of service (DoS) due to intentional attacks or because of undiscovered flaws in implementation (for example, a program written by a programmer who is unaware of a flaw that could crash the program if a certain unexpected input is encountered) * Loss of information system capabilities because of natural disasters (fires, floods, storms, or earthquakes) or human actions (bombs or strikes) * Equipment failures during normal use   Some activities that preserve confidentiality, integrity, and/or availability are granting access only to authorized personnel, applying encryption to information that will be sent over the Internet or stored on digital media, periodically testing computer system security to uncover new vulnerabilities, building software defensively, and developing a disaster recovery plan to ensure that the business can continue to exist in the event of a disaster or loss of access by personnel. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **16/6/2020** | **Name:** | **Navya R** | |
| **Course:** | **Java** | **USN:** | **4al16ec041** | |
| **Topic:** | **Natural ordering** | **Semester & Section:** | **8 A** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **REPORT** **Queue Interface In Java** The Queue interface is available in java.util package and extends the Collection interface. The queue collection is used to hold the elements about to be processed and provides various operations like the insertion, removal etc. It is an ordered list of objects with its use limited to insert elements at the end of the list and deleting elements from the start of list i.e. it follows the FIFO or the First-In-First-Out principle. Being an interface the queue needs a concrete class for the declaration and the most common classes are the [PriorityQueue](https://www.geeksforgeeks.org/priority-queue-class-in-java-2/" \t "_blank) and [LinkedList](https://www.geeksforgeeks.org/linked-list-in-java/" \t "_blank) in Java.It is to be noted that both the implementations are not thread safe. PriorityBlockingQueue is one alternative implementation if thread safe implementation is needed. Few important characteristics of Queue are:   * The Queue is used to insert elements at the end of the queue and removes from the beginning of the queue. It follows FIFO concept. * The Java Queue supports all methods of Collection interface including insertion, deletion etc. * [LinkedList](https://www.geeksforgeeks.org/linked-list-in-java/), ArrayBlockingQueue and [PriorityQueue](https://www.geeksforgeeks.org/priority-queue-class-in-java-2/" \t "_blank) are the most frequently used implementations. * If any null operation is performed on BlockingQueues, NullPointerException is thrown.   // Java orogram to demonstrate working of Queue  // interface in Java  import java.util.LinkedList;  import java.util.Queue;  public class QueueExample  {  public static void main(String[] args)  {  Queue<Integer> q = new LinkedList<>();  // Adds elements {0, 1, 2, 3, 4} to queue  for (int i=0; i<5; i++)  q.add(i);  // Display contents of the queue.  System.out.println("Elements of queue-"+q);  // To remove the head of queue.  int removedele = q.remove();  System.out.println("removed element-" + removedele);  System.out.println(q);  // To view the head of queue  int head = q.peek();  System.out.println("head of queue-" + head);  // Rest all methods of collection interface,  // Like size and contains can be used with this  // implementation.  int size = q.size();  System.out.println("Size of queue-" + size);  }  } **How to use Iterator in Java?** ‘Iterator’ is an interface which belongs to collection framework. It allows us to traverse the collection, access the data element and remove the data elements of the collection. **java.util** package has **public interface Iterator** and contains three methods:   1. **boolean hasNext()**: It returns true if Iterator has more element to iterate. 2. **Object next()**: It returns the next element in the collection until the hasNext()method return true. This method throws ‘NoSuchElementException’ if there is no next element. 3. **void remove()**: It removes the current element in the collection. This method throws ‘IllegalStateException’ if this function is called before next( ) is invoked.   // Java code to illustrate the use of iterator  import java.io.\*;  import java.util.\*;  class Test {  public static void main(String[] args)  {  ArrayList<String> list = new ArrayList<String>();  list.add("A");  list.add("B");  list.add("C");  list.add("D");  list.add("E");  // Iterator to traverse the list  Iterator iterator = list.iterator();  System.out.println("List elements : ");  while (iterator.hasNext())  System.out.print(iterator.next() + " ");  System.out.println();  }  } | | | |