**Name : Bhagyashri Jiddewar**

1. What is thread? How to implement thread in java?

Ans:-

* Basically, thread is introduced in java to overcome the situations, where we can able to run the multiple tasks simultaneously(By using the concept of time sharing concept).
* Usually, the task is divided into multiple process, and the processor can handle this process simultaneously with the help of OS, and again this piece of process also divided into smaller chunks that is called thread.
* There are two way To achieve the Threads :

1. By extending a Thread class:

* Override the run().

Example:

Object: - Thread thread=new Thread();

Class:- Class Student extended Thread{

@override

Public void run(){

}

}

1. By implementing Runnable interface:

* Override the run method.

Example:

Object:- Thread thread=new Thread(new Student);

Class:- Class Student impliments Runnable{

@override

Public void run(){

}

}

* Basically, thread is lightweight because it shares the memory space and process is heavy weight because it required its own space.
* There are two types of application

1. Single thread application.
2. Multithreaded application.

* Any program contains single thread called single threaded application.
* Any program contains multiple threads called multithreaded application.
* Each thread have its own execution path that’s why the thread of process is independent.

1. Explain life cycle of thread?

Ans:-

* Whenever the thread created it goes through some stages that is called life cycle of thread.
* There are 4-stages:

1. New thread state:

* When we initialize the thread , it will be the birth of the thread.

1. Runnable state:

* After start the thread by using start() method ,it goes into running state, here run() method can call and we can write the code for what we want to perform from thread.

1. Not Runnable state:

* When multiple threads is in the race, a processor allows only one thread at a time on the other side the all the other threads are blocked by the thread.
* The blocked threads are in the not runnable state that means it can be alive but it is in the sleep mode or rest mode with the help of wait(), sleep() method.
* whenever the lock thread release the lock the other thread can wake up from sleep mode with the help of notify() and notifyAll() method.

1. Dead State:

* After complete the execution, the thread naturally dies.
* If we want to died the thread intentionally, we need to use the destroy().

1. Explain wait() and notify() methods with example?

Ans:-

Wait() and notify() are the methods of object class

1. Wait():- This method is use to wait the thread explicitly wherever the current thread complete there execution the remaining threads can wait for the execution.
2. Notify():-This method is use to wake up the not runnable state thread or wait mode thread.
3. **Example:**

Class student{

Public static void main(String [] args){

Student student=new Student();

student.start();

Synchronized(student){

try{

System.out.println(“Waiting for the output……”);

Student.wait();

}

catch(IntrupptedException e){

System.out.println(e);

}

}

System.out.println(student.displayData());

}

Class Student

{

Public void displayData (){

Synchronized(this){

String name=”Pranay”;

Int marks=80;

System.out.println(name);

System.out.println(marks);

}

notify ();

}

}

1. What is synchronization?

Ans:-

* When multiple threads try to access single resource, it may display unexpected output, to avoid this situation we can use synchronization.
* We can synchronize the method, object or block.
* When the first thread acquires the resource it provides lock to the resource that means the thread can execute there task. After completion of the execution of 1st thread ant it release the lock that means it is open for all the threads who is on the race.

1. What is daemon thread?

Ans:-

* It is a low priority thread.
* Whenever the main thread perform the task continuously, the daemon thread works silently in the background without interrupting the main thread execution.
* It runs in background to perform task such as garbage collector and finalizer.
* If the parent thread is daemon the by default child thread is daemon.
* If parent thread is not daemon then the child is also not daemon.
* Main thread is never as a daemon thread .

1. Explain thread scheduler?

Ans:-

* It is the component in java, It is used to decide that which thread is to run or which thread is to wait that is called thread schedular.
* At a time number of threads are in the running state, that time schedular can decide which tread can run or which one is wait.
* There are two way to decide:

1. Priority:-

* If the thread has higher priority, it may have more chance to pick by time schedular.

1. Time Slicing:-

* If the two threads are in the running state with same priority, that time it will be decided from first come first serve algorithm.
* If the first thread takes more time to complete the execution, it goes into the infinite blocking to avoid this situation, the time slicing concept came.
* Whenever this happens the CPU gives the permission to the another thread to perform the execution, that is the preemptive priority Scheduling.

1. Write a program to write object to file and read object from file

Ans:-

1. Write a Java program to find the longest word in a text file.

Ans:-

1. Write a Java program to read the first 3 lines of a file.

Ans:-

package Test\_2;  
  
import java.io.BufferedReader;  
import java.io.FileNotFoundException;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class ReadFirstThreeLine {  
 public static void displayFirstThreeLine(){  
 try {  
 BufferedReader bufferedReader = new BufferedReader(new FileReader("nik.txt"));  
 String lineRead = bufferedReader.readLine();  
 while (((lineRead!= null))) {  
 for(int i=0;i<3;i++)  
 System.*out*.println("First three lines : "+lineRead);  
 lineRead = bufferedReader.readLine();  
 }  
 } catch (FileNotFoundException e) {  
 throw new RuntimeException(e);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 }  
}

package Test\_2;  
import java.io.\*;  
  
import java.io.IOException;  
  
public class Main {  
  
 public static void main(String[] args){  
  
// ======================== Append text to existing file (WriteObjectReadObject)=================================  
  
// WriteObjectReadObject.displayData();  
  
// ========================= Store data line by line (TextFileArrayLineByLine)===================================  
  
// TextFileArrayLineByLine.displayArrayDataLineByLine();  
  
// ========================== Read First Three Line of file data ================================================  
  
 ReadFirstThreeLine.*displayFirstThreeLine*();  
 }  
}

1. Write a Java program to append text to an existing file

Ans:-

package Test\_2;  
  
import java.io.\*;  
public class WriteObjectReadObject {  
 public static void displayData(){  
  
 try {  
 PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("nik123.txt", true)));  
 out.println("the text");  
 out.close();  
 BufferedReader bufferedReader=new BufferedReader(new FileReader("nik123.txt"));  
 String str=bufferedReader.readLine();  
 System.*out*.println("Existing + Additional data :");  
 while(str!=null) {  
 System.*out*.println(str +"\n");  
 str=bufferedReader.readLine();  
 }  
  
 } catch (IOException e) {  
 System.*out*.println(e);  
 }  
  
 }  
}

package Test\_2;  
import java.io.\*;  
  
import java.io.IOException;  
  
public class Main {  
  
 public static void main(String[] args){  
  
// ======================== Append text to existing file =====================================  
 WriteObjectReadObject.*displayData*();  
// ============================================================  
 }  
}

1. Write a Java program to store text file content line by line in an array.

Ans:-

package Test\_2;  
  
import java.io.\*;  
  
public class TextFileArrayLineByLine {  
 public static void displayArrayDataLineByLine(){  
  
 try {  
 BufferedReader in = new BufferedReader(new FileReader("nik.txt"));  
 String str;  
 while ((str = in.readLine()) != null) {  
 String[] arr=new String[str.length()];  
 for (int i = 0; i < str.length(); i++) {  
 arr[i] = in.readLine();  
 System.*out*.println("data = "+arr[i]);  
 }  
 }  
 } catch (FileNotFoundException e) {  
 throw new RuntimeException(e);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
  
 }  
}

package Test\_2;  
import java.io.\*;  
  
import java.io.IOException;  
  
public class Main {  
  
 public static void main(String[] args){  
  
// ======================== Append text to existing file (WriteObjectReadObject)=====================================  
// WriteObjectReadObject.displayData();  
// ========================= Store data line by line (TextFileArrayLineByLine)===================================  
 TextFileArrayLineByLine.*displayArrayDataLineByLine*();  
 }  
}