UNIT –IV

1. What are the benefits of Multithreading?

- Multithreading increases the **responsiveness** of system as, if one thread of the application is not responding, the other would respond in that sense the user would not have to sit idle.
- Multithreading allows **resource sharing** as threads belonging to the same process can share code and data of the process and it allows a process to have multiple threads at the same time active in **same address space**.
- Creating a different process is costlier as the system has to allocate different memory and resources to each process, but creating threads is easy as it does not require allocating separate memory and resources for threads of the same process.

2. What are the differences between Multithreading and Multitasking?

Parameter	Multithreading	Multitasking
Definition	Multithreading is to execute multiple threads in a process concurrently.	Multitasking is to run multiple processes on a computer concurrently.
execution	In Multithreading, the CPU switches between multiple threads in the same process.	In Multitasking, the CPU switches between multiple processes to complete the execution.
resource sharing	In Multithreading, resources are shared among multiple threads in a process.	In Multitasking, resources are shared among multiple processes.
Complexity	Multithreading is light-weight and easy to create.	Multitasking is heavy-weight and harder to create.

3. Define thread.

A thread is a basic execution unit which has its own program counter, set of the register and stack. But it shares the code, data, and file of the process to which it belongs.

4. List the states of thread life cycle.

A thread at any point of time exists in any one of the following states.

1. New
2. Runnable
3. Blocked
4. Waiting
5. Timed Waiting
6. Terminated

5. List some methods supported by threads.

- ***join():*** It makes to wait for this thread to die. We can wait for a thread to finish by calling its join() method.
- ***sleep():*** It makes current executing thread to sleep for a specified interval of time. Time is in milli seconds.

- *yield():* It makes current executing thread object to pause temporarily and gives control to other thread to execute.
- ***notify():*** This method is inherited from Object class. This method wakes up a single thread that is waiting on this object's monitor to acquire lock.
- ***notifyAll():*** This method is inherited from Object class. This method wakes up all threads that are waiting on this object's monitor to acquire lock.
- ***wait():*** This method is inherited from Object class. This method makes current thread to wait until another thread invokes the notify() or the notifyAll() for this object.

6. Can we start a thread twice?

No. After starting a thread, it can never be started again. If we do so, an Illegal Thread-State Exception is thrown. In such case, thread will run once but for second time, it will throw exception.

7. What is the use of join() method?

The join() method waits for a thread to die. In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task.

Syntax:

public void join()throws InterruptedException
public void join (long milliseconds) throws Interrupted Exception

8. Why do we need Synchronization?

The synchronization is mainly used to

i) To prevent thread interference.

ii) To prevent consistency problem.

9. What are the types of Synchronization?

There are two types of synchronization:

i) Process Synchronization

ii) Thread Synchronization

10. Write about Java synchronized method.

- If we declare any method as synchronized, it is known as synchronized method.
- Synchronized method is used to lock an object for any shared resource.
- When a thread invokes a synchronized method, it automatically acquires the lock for that object and releases it when the thread completes its task.

11. What do you mean by Synchronized block in java?

Synchronized block can be used to perform synchronization on any specific resource of the method. Suppose we have 50 lines of code in your method, but we want to synchronize only 5 lines, we can use synchronized block.
If we put all the codes of the method in the synchronized block, it will work same as the synchronized method.

12. What is synchronization? Briefly explain.

Two or more threads accessing the same data simultaneously may lead to loss of data integrity. For example, when two people access a savings account, it is possible that one person may overdraw and the cheque may bounce. The importance of updating of the pass book can be well understood in this case.

13. Why would you use a synchronized block vs. synchronized method?

Synchronized blocks place locks for shorter periods than synchronized methods.

14. What's the difference between the methods sleep() and wait()

The code sleep(1000); puts thread aside for exactly one second. The code wait(1000), causes a wait of up to one second. A thread could stop waiting earlier if it receives the notify() or notifyAll() call. The method wait() is defined in the class Object and the method sleep() is defined in the class Thread.

15. What is synchronization and why is it important?

With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object's value. This often leads to significant errors.

16. What are three ways in which a thread can enter the waiting state?

A thread can enter the waiting state by invoking its sleep() method, by blocking on I/O, by unsuccessfully attempting to acquire an object's lock, or by invoking an object's wait() method. It can also enter the waiting state by invoking its (deprecated) suspend() method.

17. When you will synchronize a piece of your code?

When you expect your code will be accessed by different threads and these threads may change a particular data causing data corruption.

18. What is daemon thread and which method is used to create the daemon thread?

Daemon thread is a low priority thread which runs intermittently in the back ground doing the garbage collection operation for the java runtime system. setDaemon method is used to create a daemon thread.

19. What is the difference between yielding and sleeping?

When a task invokes its yield() method, it returns to the ready state. When a task invokes its sleep() method, it returns to the waiting state.

20. Can I implement my own start() method?

The Thread start() method is not marked final, but should not be overridden. This method contains the code that creates a new executable thread and is very specialised. Your threaded application should either pass a Runnable type to a new Thread, or extend Thread and override the run() method.

21. Do I need to use synchronized on setValue(int)?

It depends whether the method affects method local variables, class static or instance variables. If only method local variables are changed, the value is said to be *confined* by the method and is not prone to threading issues.

22. What is thread priority?

Thread Priority is an integer value that identifies the relative order in which it should be executed with respect to others. The thread priority values ranging from 1- 10 and the default value is 5. But if a thread have higher priority doesn't means that it will execute first. The thread scheduling depends on the OS.

23. What are the different ways in which a thread can enter into waiting state?

There are three ways for a thread to enter into waiting state. By invoking its sleep() method, by blocking on I/O, by unsuccessfully attempting to acquire an object's lock, or by invoking an object's wait() method.

24. How would you implement a thread pool?

The Thread Pool class is a generic implementation of a thread pool, which takes the following input Size of the pool to be constructed and name of the class which implements Runnable (which has a visible default constructor) and constructs a thread pool with active threads that are waiting for ctivation. once the threads have finished processing they come back and wait once again in the pool.

25. What is Generics? List its advantages.

Generics are similar to templates in C++. Using Generics, the programmer can create a single class, interface or method that automatically works with all types of data (Integer, String, Float etc) i.e. the code can be reused for object of many different types.

Advantages of Generics:

- Generics make the code safer and easier to read.
- Type casting is not needed.
- It is checked at compile time. So problem will not be generated at run time.

26. How to implement generic method in Java?

Generic methods are methods that can accept any type of argument.

Syntax:

<type-parameter> return_type method_name (parameters) {...}

Example:

```
public void display(T data) {
    System.out.println(data);
}
```

27. What is the question mark in Java Generics used for?

The wildcard element is represented by ? and it specifies an unknown type i.e. it means any type. For example,

<? extends Number>

specifies any child classes of class Number e.g. Integer, Float, double etc.

28. What do you mean by bounded type parameters?

In Generics, bounded type parameter set restriction on the type that will be allowed to pass to a type-parameter.

Syntax: <T extends superclass>

For example, <T extends Number>

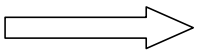
In this example, the type parameter is restricted to any child classes of class Number e.g. Integer, Float, double etc.

29. What is Erasure?

The Java compiler applies type erasure to implement generic programming. It replaces all type parameters in generic types with their bounds or Object if the type parameters are unbounded.

Example: Java class with Generics

```
class Gen<T>{
        T get(){
        …}
}
```



Example: T replaced by java.lang.Object

```
class Gen extends java.lang.Object{
        java.lang.Object get(){
        …}
}
```

30. List out the restrictions to be considered in generic programming.

- Cannot Instantiate Generic Types with Primitive Types
- Cannot Create Instances of Type Parameters
- Cannot Declare Static Fields Whose Types are Type Parameters
- Cannot Use Casts or instanceof With Parameterized Types
- Cannot Create Arrays of Parameterized Types
- Cannot Create, Catch, or Throw Objects of Parameterized Types

Part-B

1. With a simple program explain the multithreading concept in Java.(16)

(CS1261 NOV /DEC 2016) (CS2311 APR/MAY-2015)

2. Write a complex program to illustrate about thread priorities. Imagine that the first thread has just begun to run, even before it has a chance to do anything. Now the higher priority thread that wants to run as well. Now the higher priority thread has to do its work before the first thread starts. (16). (IT2301 APR/MAY-2015)
3. Explain life cycle of a thread with help of a diagram. How do the wait and notify all/no-tify methods enable cooperation between thread? (8) (IT2301 APR/MAY-2015)
4. Explain in detail about generic method with a suitable example. (8)

(IT2301 APR/MAY-2015)

5. With illustrations explain multithreading, interrupting threads, thread states and thread priorities. (16) (IT2301 MAY/JUNE-2014)
6. What is a Thread? What are the different states of Thread? Explain the creation of Thread with an example program. (16) (CS1361 NOV/DEC-2014)
7. Using generic classes, write a program to perform the following operations on an array.

(IT2301 MAY/JUNE-2014)

a. Add an element in the beginning/middle/end. (8)

b. Delete an element from a given position. (8)

8. What are interrupting threads? Explain thread states and synchronization. (16)

(IT2301 NOV/DEC-2012)

9. What is multithreading? Explain the two methods of implementing threads with an example. (16)(CS2311 NOV/DEC-2014)

10. Explain the lifecycle of a thread in detail with example program.

(CS2311 MAY/JUNE-2014)