OBJECT ORIENTED PROGRAMMING WITH JAVA(21CS35)
UNIT 3: PRACTICE QUESTIONS

1. Discuss exception handling statements of Java with syntax and examples 6m
2. Distinguish between checked and unchecked exceptions 5m
3. Devise a Java program to handle divide by zero error 5m
4. Devise a Java program to handle ArithmeticException and ArrayIndexOutOfBoundsException depending on the number of arguments passed to the main 6m
5. Illustrate nested try/catch with example 6
6. Devise a Java program to handle NullPointerException 6m
7. Distinguish between throw and throws 5m
8. Demonstrate throws with syntax and an example 6m
9. Demonstrate finally with syntax and an example 6m
10. Devise a Java program to handle IllegalAccess exception 6m
11. Devise a Java program to handle NumberFormatException 6m
12. Discuss the creation of custom exception classes in Java with syntax of necessary constructs and an example 8m
13. Devise a Java program to add 2 objects of Distance Class. Distance class has 2 instance variables-feet and inches.If the inches entered from the console is >= 12 throw an exception of Custom exception class and catch it to display "Exception Raised---Inches value should be <12 , you entered it as : ", inches and, exit the program. 10m
14. Discuss the creation of chained exception classes in Java with syntax of necessary constructs and an example. 10m
15. Devise a Java program to handle a chained exception. The top layer exception is NullPointerException and the cause exception is ArithmeticException 8m
16. Compare Process-based and Thread-based multitasking 5m
17. Discuss any 5 Advantages of multithreading 5m
18. Outline the major aspects of the Java Thread model 6m
19. Devise a Java program to change the name and priority of the main thread. Main thread Displays numbers from 1 to 100 . Show the name, priority and thread pool name before and after the changes. 6m
20. Write syntax of sleep() and give an example to illustrate it's use 6m
21. Discuss the process of creating threads in Java by implementing Runnable Interface. Give syntax of necessary constructs and a program example. 8m
22. Discuss the process of creating threads in Java by extending Thread class. Give syntax of necessary constructs and a program example. 8m
23. Demonstrate the process of creating multiple threads in Java that share the common run method with help of an example program. 8m
24. The venue of an event has a seating capacity of 50. Tickets need to be cut and the seat number need to be shown for 50 seats. Write a multi-threaded Java program to spawn 2 threads from main to share the same run method. One thread must show the message :"Cut the ticket", ticketno. Other must show the message:" Show your seat number", seatno. 8m
25. Demonstrate the use of isAlive() in Java with help of a suitable example program. 8m
26. Develop a java program to spawn 2 threads from main. One thread generates odd numbers from 1 to 10 and finds sum. Other thread generates even numbers from 1 to 10 and finds sum. The main thread should wait for the threads to complete and then find total sum of odd and even numbers. Use join() to wait for threads to finish.
27. Write a java program to spawn 2 threads from main. One thread generates prime numbers from 1 to 50 and finds sum. Other thread generates fibonacci series numbers from 1 to 50 and finds sum. Set the priority of Prime and Fibonacci threads to 10 and 2 respectively.
28. Demonstrate synchronization of threads in Java with help of a suitable example program. 8m

OBJECT ORIENTED PROGRAMMING WITH JAVA(21CS35)
UNIT 3: PRACTICE QUESTIONS

29. Write a java program to spawn 2 threads from main. The spawned threads will generate prime numbers from 1 to 50 by calling the method void generatePrimenumbers() of another class namely generatePrime. Prevent the 2 threads from racing to complete generatePrimenumbers() by declaring it as synchronized.
30. Develop a Java program to implement a simple form of the producer/consumer problem. It consists of four classes: Q, the queue that you're trying to synchronize; Producer, the threaded object that is producing queue entries; Consumer, the threaded object that is consuming queue entries; and PC, the tiny class that creates the single Q, Producer, and Consumer.(use wait() and notify() for appropriate synchronization of the threads)
31. Develop a Java program to illustrate deadlock where two threads compete to acquire lock on the two synchronized objects.