********** Java Revision Plan **********
TOTAL APPROX. QUESTION 400

Day 1-2: Variables and Literals (30 Questions)

1. Write a program to declare and initialize variables of all primitive data types in Java.
2. Demonstrate the use of constant literals using `final` keyword.
3. Write a program to swap two variables without using a third variable.
4. Write a program to find the sum of two numbers entered by the user.
5. Differentiate between local, instance, and static variables using examples.
6. Implement a program to calculate the area of a rectangle using variables.
7. Write a program to demonstrate type inference with `var`.
8. Declare variables for different units of a product and calculate the total cost.
9. Create a program to store and display the ASCII values of characters.
10. Show the difference between variable initialization and declaration.
11. Write a program to demonstrate overflow in byte variables.
12. Create a program to assign and print hexadecimal and binary literals.
13. Demonstrate the use of underscores in numeric literals.
14. Write a program to store and print the value of π (pi).
15. Implement a program using boolean literals.
16. Create a program to demonstrate default values of instance variables.
17. Write a program to calculate compound interest.
18. Use variables to store user profile information and display it.
19. Write a program to calculate the perimeter of a triangle.
20. Create a program that uses escape sequences in string literals.
21. Write a program to initialize and manipulate a character array.
22. Implement a program for arithmetic operations using variables.
23. Write a program to print a pyramid pattern using string literals.
24. Write a program to calculate the Body Mass Index (BMI).
25. Demonstrate the use of `null` as a literal.
26. Create a program to calculate electricity bill based on consumption.
27. Write a program to declare multiple variables in a single statement.
28. Demonstrate variable shadowing in nested blocks.
29. Write a program to display the multiplication table of a number.
30. Create a program to store and display a date using variables.

Day 5-6: Operators and Operations (30 Questions)

1. Write a program to demonstrate arithmetic operators.
2. Implement a program using relational operators.
3. Demonstrate logical operators with a truth table.
4. Write a program to use bitwise operators.
5. Create a program to check if a number is even or odd using the modulo operator.
6. Write a program to swap two numbers using XOR operator.
7. Demonstrate the use of the ternary operator.
8. Write a program to check if a number is a power of 2 using bitwise operators.
9. Create a program to use shift operators.
10. Write a program to increment and decrement variables using unary operators.
11. Implement a calculator using switch-case and arithmetic operators.
12. Write a program to find the maximum of three numbers using conditional operators.
13. Create a program to use the assignment operator.
14. Demonstrate the use of instanceof with objects.
15. Write a program to perform operations on floating-point numbers.
16. Implement a program to calculate the square root using Math class.

17. Create a program to check if a number is prime using loops and logical operators.
18. Write a program to demonstrate short-circuit logical operators.
19. Implement a program using compound assignment operators.
20. Write a program to find the absolute value of a number using Math.abs().
21. Demonstrate precedence and associativity of operators in a program.
22. Write a program to perform type promotion in expressions.
23. Create a program to count the set bits in an integer using bitwise operators.
24. Write a program to demonstrate integer division and floating-point division.
25. Implement a program to find the GCD of two numbers using modulus operator.
26. Create a program to use the increment operator in a loop.
27. Write a program to calculate the area of a circle using Math.PI.
28. Demonstrate the use of Math.pow() to calculate exponents.
29. Write a program to compare strings using relational operators and equals method.
30. Implement a program to toggle the case of a character using bitwise XOR.

Day 3-4: Upcasting and Downcasting (30 Questions)

1. Write a program to demonstrate upcasting with class inheritance.
2. Implement a program to perform downcasting and handle `ClassCastException`.
3. Write a program to cast a `double` to `int` and observe the loss of precision.
4. Demonstrate the use of upcasting in polymorphism.
5. Implement a program where upcasting helps simplify method overriding.
6. Write a program to convert a `long` to `short` using casting.
7. Create an example where upcasting is applied to an interface.
8. Implement a program to store subclass objects in a superclass array.
9. Write a program to demonstrate explicit type casting.
10. Demonstrate `instanceof` to safely downcast objects.
11. Create a program to cast a `float` to `int` and observe the results.
12. Write a program to demonstrate upcasting with abstract classes.
13. Implement a program to compare the outputs of upcasting and downcasting.
14. Demonstrate upcasting in a collection like `ArrayList`.
15. Write a program to use downcasting in a `switch` case scenario.
16. Implement a program where upcasting restricts access to specific methods.
17. Write a program to cast an `int` to `char` and display the character.
18. Create a program to demonstrate narrowing conversion.
19. Write a program to cast a parent class to a child class and access child-specific methods.
20. Demonstrate upcasting using a real-world example (e.g., vehicle types).
21. Create a program to show how upcasting works with method overriding.
22. Write a program to demonstrate downcasting with multiple levels of inheritance.
23. Implement a program to cast a wrapper class object to a primitive type.
24. Write a program to upcast an object and call a method of the superclass.
25. Create an example where downcasting is needed in a generic collection.
26. Demonstrate the loss of data when downcasting a higher precision type to a lower precision one.
27. Write a program to upcast and downcast using enums.
28. Implement a program that uses upcasting for function arguments.
29. Create a program to perform downcasting in a class hierarchy.
30. Write a program to use casting with numeric literals.

Day 7-8: Scanner and BufferReader (30 Questions)

1. Write a program to read an integer from the user using `Scanner`.
2. Implement a program to read a floating-point number using `Scanner`.
3. Write a program to read a string input from the user using `Scanner`.
4. Create a program to demonstrate the use of `nextLine()` in `Scanner`.

5. Write a program to read multiple inputs of different types using `Scanner`.
6. Implement a program to add two numbers using input from `Scanner`.
7. Write a program to read an integer array from the user using `Scanner`.
8. Create a program to read a character input using `Scanner`.
9. Write a program to calculate the sum of integers entered by the user until a negative number is encountered.
10. Implement a program to read and reverse a string using `Scanner`.
11. Write a program to read user details like name, age, and address using `Scanner`.
12. Demonstrate how to handle `InputMismatchException` while reading inputs with `Scanner`.
13. Write a program to read a double value and format it to two decimal places.
14. Implement a program to read space-separated integers from the user.
15. Write a program to read a sentence and count the number of words using `Scanner`.
16. Create a program to read a file line by line using `Scanner`.
17. Write a program to read inputs for a matrix and print its transpose.
18. Implement a program to check if a given input string is a palindrome.
19. Write a program to demonstrate `Scanner` vs `BufferedReader` performance.
20. Create a program to read input using `BufferedReader` and `InputStreamReader`.
21. Write a program to read and sum integers from a file using `Scanner`.
22. Implement a program to read user input until "STOP" is entered using `BufferedReader`.
23. Write a program to read a large paragraph of text using `BufferedReader`.
24. Create a program to read multiple inputs separated by commas using `BufferedReader`.
25. Write a program to read and parse date input using `BufferedReader`.
26. Implement a program to read a line and split it into words using `BufferedReader`.
27. Write a program to read two strings and concatenate them using `BufferedReader`.
28. Create a program to compare two inputs read using `Scanner` and `BufferedReader`.
29. Write a program to demonstrate exception handling while using `BufferedReader`.
30. Implement a program to read an input and check if it's numeric using `BufferedReader`.

Day 9-10: String Concepts (30 Questions)

1. Write a program to compare two strings using equals().
2. Implement a program to demonstrate StringBuilder for string concatenation.
3. Write a program to find the length of a string.
4. Create a program to check if a string is a palindrome.
5. Write a program to reverse a string using StringBuilder.
6. Implement a program to count the number of vowels in a string.
7. Write a program to convert a string to uppercase and lowercase.
8. Create a program to replace a substring within a string.
9. Write a program to split a sentence into words.
10. Implement a program to find the index of a character in a string.
11. Write a program to demonstrate StringBuffer methods.
12. Write a program to check if a string contains only digits.
13. Create a program to remove all spaces from a string.
14. Implement a program to join multiple strings using String.join().
15. Write a program to find the first non-repeating character in a string.
16. Create a program to count occurrences of a character in a string.
17. Implement a program to compare two strings lexicographically.
18. Write a program to demonstrate the use of String.format().
19. Create a program to check if a string starts with a given prefix.
20. Write a program to extract a substring from a string.
21. Implement a program to find all permutations of a string.
22. Create a program to reverse each word in a sentence.
23. Write a program to demonstrate the use of trim() method.

24. Implement a program to sort an array of strings.
25. Write a program to find common characters between two strings.
26. Create a program to demonstrate String.intern() method.
27. Implement a program to compare string literals and string objects using == and equals().
28. Write a program to find the longest common prefix among an array of strings.
29. Create a program to remove duplicate characters from a string.
30. Write a program to demonstrate the immutability of strings in Java.

Day 11-12: Conditions (30 Questions)

1. Write a program to check if a number is positive, negative, or zero.
2. Implement a program to find the largest of three numbers using nested if-else.
3. Write a program to determine whether a year is a leap year.
4. Create a program to categorize an age into child, teenager, or adult.
5. Write a program to determine whether a character is a vowel or consonant.
6. Implement a program to calculate grade based on marks using if-else.
7. Write a program to check if a number is divisible by both 5 and 11.
8. Create a program to find the day of the week based on a number using if-else.
9. Write a program to check if a number is even or odd using conditional statements.
10. Implement a program to categorize a character as uppercase, lowercase, or digit.
11. Write a program to check if a string contains only alphabets.
12. Create a program to implement a simple calculator using if-else.
13. Write a program to determine whether a number is prime using conditions.
14. Implement a program to find the roots of a quadratic equation.
15. Write a program to determine if a triangle is valid based on its angles.
16. Create a program to check if a person is eligible to vote.
17. Write a program to classify a number as positive, negative, or zero using a ternary operator.
18. Implement a program to determine whether a number falls within a range.
19. Write a program to calculate the electricity bill based on consumption slabs.
20. Create a program to classify a person based on Body Mass Index (BMI).
21. Write a program to validate a password input.
22. Implement a program to check if a given year is a century year.
23. Write a program to find the smallest of three numbers using conditional statements.
24. Create a program to determine the type of a given character (special, digit, or letter).
25. Write a program to implement a menu-driven program using if-else.
26. Implement a program to check whether a number is a palindrome.
27. Write a program to calculate tax based on income slabs.
28. Create a program to determine if a number is an Armstrong number using conditions.

29. Write a program to classify a number based on its last digit.
30. Implement a program to check if a number is perfect or not.

Day 13-14: Loops (30 Questions)

1. Write a program to print numbers from 1 to 100 using a loop.
2. Implement a program to find the sum of the first N natural numbers.
3. Write a program to print the multiplication table of a number.
4. Create a program to reverse a given number using a loop.
5. Write a program to check if a number is a palindrome using a loop.
6. Implement a program to find the factorial of a number.
7. Write a program to generate Fibonacci series up to N terms.
8. Create a program to count the digits of a given number.
9. Write a program to calculate the sum of digits of a number.
10. Implement a program to print all prime numbers within a range.

Write a program to display the pattern:
*

**

***

****

11. *****
12. Create a program to check whether a number is prime.
13. Write a program to find the greatest common divisor (GCD) of two numbers using loops.
14. Implement a program to print numbers in reverse order.
15. Write a program to find the sum of even numbers between 1 and 100.
16. Create a program to display numbers in a pyramid pattern.
17. Write a program to calculate the power of a number using a loop.
18. Implement a program to check whether a number is an Armstrong number.
19. Write a program to find the LCM of two numbers using loops.
20. Create a program to calculate the sum of squares of the first N natural numbers.
21. Write a program to print the ASCII values of characters from A to Z.
22. Implement a program to find the sum of a series: 1 + 1/2 + 1/3 + ... + 1/N.
23. Write a program to print all factors of a number using loops.

Create a program to display the following pattern:
1

121

24. 12321
25. Write a program to find the smallest and largest digit in a number.
26. Implement a program to print the reverse of a string using loops.

Write a program to display the pattern:
1

22

333

27. 4444
28. Create a program to check if a number is a perfect number.
29. Write a program to generate and print Pascal's Triangle.
30. Implement a program to count the occurrence of a digit in a number.

Day 15-16: Arrays (30 Questions)

1. Write a program to initialize and print elements of a one-dimensional array.
2. Implement a program to calculate the sum of elements in an array.
3. Write a program to find the largest element in an array.
4. Create a program to reverse an array.
5. Write a program to find the second largest element in an array.
6. Implement a program to perform linear search in an array.
7. Write a program to sort an array using bubble sort.
8. Create a program to merge two arrays.
9. Write a program to remove duplicate elements from an array.
10. Implement a program to find the frequency of elements in an array.
11. Write a program to calculate the average of elements in an array.
12. Create a program to perform binary search on a sorted array.
13. Write a program to find the intersection of two arrays.
14. Implement a program to rotate an array to the left by one position.
15. Write a program to find the maximum sum of a subarray (Kadane's Algorithm).
16. Create a program to check if an array is sorted.
17. Write a program to calculate the product of all elements in an array.
18. Implement a program to find the missing number in a sequence.
19. Write a program to find the minimum distance between two numbers in an array.
20. Create a program to split an array into two halves.
21. Write a program to find all pairs in an array with a given sum.
22. Implement a program to find the union of two arrays.
23. Write a program to transpose a 2D array (matrix).
24. Create a program to perform addition of two matrices.
25. Write a program to find the row with the maximum sum in a 2D array.

26. Implement a program to sort the rows of a matrix.
27. Write a program to count the number of even and odd elements in an array.
28. Create a program to reverse the diagonal elements of a 2D array.
29. Write a program to implement selection sort on an array.
30. Implement a program to find the largest square sub-matrix with all 1s in a binary matrix.

Day 17-18: Classes and Objects (30 Questions)

1. Write a program to create a class Student with attributes name, roll number, and marks, and methods to input and display details.
2. Create a class Rectangle with methods to calculate area and perimeter. Instantiate objects and perform operations.
3. Implement a class BankAccount with attributes like account number, holder name, and balance. Include methods to deposit and withdraw money.
4. Write a program to create a class Car with attributes like model, brand, and price, and a method to display the car details.
5. Create a class Book to store details like title, author, and price, and implement methods for discounts.
6. Design a class Employee with methods to calculate gross and net salary based on basic salary and allowances.
7. Write a program to create a class Circle with a method to calculate area and circumference.
8. Implement a class Complex to add, subtract, and multiply two complex numbers.
9. Write a program to create a class Student with a constructor for initializing attributes and a method to calculate the grade.
10. Design a class Movie to store title, director, and rating, and include methods to display details.
11. Create a class Calculator with static methods for addition, subtraction, multiplication, and division.
12. Implement a class Date with methods to compare two dates and find the difference in days.
13. Write a program to create a class Shape with a method to calculate area (method to be overridden in derived classes).
14. Create a class Product with methods to apply discounts and calculate final price.
15. Write a program to create a class Bank with methods to calculate compound interest and simple interest.
16. Design a class Vehicle with methods to calculate fuel efficiency and maintenance cost.
17. Implement a class Account with methods for deposit, withdrawal, and balance inquiry.

18. Write a program to create a class Person with attributes name and age, and implement a method to check voter eligibility.
19. Create a class Library with methods to issue, return, and check availability of books.
20. Write a program to create a class Timer to measure execution time of code snippets.
21. Implement a class Triangle with methods to check validity and calculate area.
22. Write a program to create a class Queue with methods for enqueue and dequeue operations.
23. Design a class Stack with methods for push, pop, and peek operations.
24. Create a class Contact to store and display contact details of a person.
25. Write a program to implement method overloading in a class MathOperations.
26. Implement a class Currency to perform addition and subtraction of amounts in different denominations.
27. Write a program to create a class Reservation with methods to book and cancel tickets.
28. Create a class Temperature to convert between Celsius and Fahrenheit.
29. Write a program to create a class Inventory to manage stock details of a store.
30. Implement a class StudentDatabase to perform CRUD (Create, Read, Update, Delete) operations.

Day 19-20: OOP Concepts (30 Questions)

Encapsulation

1. Write a program to demonstrate encapsulation with a class Account.
2. Create a program to demonstrate data hiding and access specifiers.
3. Write a program to demonstrate the use of this keyword in constructor and method calls.
4. Implement a program to create a singleton class.
5. Write a program to demonstrate object cloning using Cloneable interface.
6. Create a program to demonstrate static and non-static blocks in a class.
7. Write a program to demonstrate the final keyword with variables, methods, and classes.
8. Implement a program to create immutable objects in Java.
9. Write a program to demonstrate how to validate data using encapsulation principles.
10. Create a program to demonstrate nested classes and their usage.

Inheritance

11. Write a program to demonstrate inheritance with a base class Animal and derived classes Dog and Cat. 12. Create a program to show the "is-a" and "has-a"

relationships in classes. 13. Write a program to demonstrate constructor chaining in inheritance. 14. Implement a program to demonstrate the use of super keyword for method and constructor calls. 15. Create a program to demonstrate multiple levels of inheritance. 16. Write a program to demonstrate downcasting and upcasting in inheritance. 17. Implement a program to demonstrate the use of instanceof operator in inheritance. 18. Write a program to demonstrate the use of covariant return types. 19. Create a program to show how default constructors are used in inheritance. 20. Implement a program to demonstrate method overriding in inheritance.

Polymorphism

21. Implement polymorphism using method overriding in a class hierarchy. 22. Write a program to demonstrate method overloading in a class MathOperations. 23. Create a program to show dynamic method dispatch. 24. Write a program to implement runtime polymorphism using an interface. 25. Create a program to demonstrate the use of abstract and concrete classes. 26. Implement a program to demonstrate the Template Method design pattern using abstract classes. 27. Write a program to demonstrate polymorphic behavior with collections. 28. Create a program to implement operator overloading using method overloading. 29. Write a program to demonstrate how Java handles polymorphic references. 30. Implement a program to demonstrate the factory design pattern.

Abstraction

31. Create an abstract class Shape with derived classes Circle, Rectangle, and Triangle implementing the abstract methods. 32. Write a program to demonstrate interface implementation and multiple inheritance. 33. Implement a program to demonstrate the use of abstract keyword with methods. 34. Write a program to demonstrate the difference between an abstract class and an interface. 35. Create a program to implement a payment gateway system using interfaces. 36. Implement a program to demonstrate how to use functional interfaces. 37. Write a program to demonstrate the use of lambda expressions with an interface. 38. Create a program to implement a callback mechanism using interfaces. 39. Write a program to show the benefits of abstraction in reducing code complexity. 40. Implement a program to demonstrate loose coupling using abstraction.

Day 21: Interfaces (30 Questions)

1. Create an interface Animal with methods eat and sleep, and implement it in classes Dog and Cat.
2. Write a program to demonstrate multiple inheritance using interfaces.
3. Create an interface Shape with a method calculateArea and implement it in classes Circle and Rectangle.
4. Implement a program where an interface is used to define a callback mechanism.
5. Create an interface Vehicle with methods start and stop. Implement it in classes Car and Bike.

6. Write a program to demonstrate a functional interface using @FunctionalInterface annotation.
7. Implement a program to create a default method in an interface and override it in the implementing class.
8. Write a program to demonstrate the use of static methods in an interface.
9. Create an interface Payment with a method processPayment and implement it in classes CreditCard and DebitCard.
10. Write a program to demonstrate how multiple interfaces can be implemented by a single class.
11. Create a program where one interface extends another interface.
12. Implement a program to use Comparator interface to sort a list of objects.
13. Write a program to demonstrate the use of Iterable interface in a custom collection class.
14. Create a program to implement a calculator using an interface.
15. Write a program to demonstrate marker interfaces and their purpose.
16. Implement a program to demonstrate the use of Runnable interface in a multithreading context.
17. Write a program to demonstrate loosely coupled architecture using interfaces.
18. Create an interface MediaPlayer and implement it in classes VLCPlayer and WindowsMediaPlayer.
19. Write a program to demonstrate the concept of dependency injection using interfaces.
20. Implement a program where an interface is used to represent a collection of constants.
21. Create a program to demonstrate dynamic method dispatch using interfaces.
22. Write a program to demonstrate how interfaces are used in callback mechanisms.
23. Implement a program to demonstrate the Comparable interface in sorting objects.
24. Write a program to demonstrate the Cloneable interface with deep and shallow copying.
25. Create an interface EventListener and implement it in classes ButtonClickListener and WindowCloseListener.
26. Write a program to demonstrate the use of Map.Entry interface in Java collections.
27. Implement a program to demonstrate the use of adapter design pattern using interfaces.
28. Write a program to demonstrate the use of bridge design pattern with interfaces.
29. Create a program to implement a factory design pattern using interfaces.
30. Write a program to demonstrate the use of lambda expressions in interfaces.

Day 22: Lambda Expressions (30 Questions)

1. Write a program to implement a lambda expression for sorting a list of strings by length.
2. Create a program to demonstrate the use of lambda expressions with Runnable interface.
3. Write a program to filter a list of numbers using lambda expressions.
4. Implement a program to demonstrate the use of lambda expressions with Comparator interface.
5. Write a program to use a lambda expression to calculate the sum of two numbers.

6. Create a program to demonstrate the use of lambda expressions in a custom functional interface.
7. Write a program to iterate through a list of integers using lambda expressions.
8. Implement a program to use lambda expressions to check if a string starts with a given letter.
9. Write a program to demonstrate the use of lambda expressions with streams for filtering and mapping.
10. Create a program to use lambda expressions to find the maximum element in a list.
11. Write a program to demonstrate the use of lambda expressions in event handling.
12. Implement a program to sort a list of objects using lambda expressions and a custom comparator.
13. Write a program to demonstrate the use of lambda expressions with the Consumer functional interface.
14. Create a program to demonstrate the use of lambda expressions with the Predicate functional interface.
15. Write a program to demonstrate the use of lambda expressions with the Function functional interface.
16. Implement a program to demonstrate chaining of Predicate interfaces using lambda expressions.
17. Write a program to demonstrate the use of lambda expressions with BiFunction functional interface.
18. Create a program to demonstrate the use of lambda expressions with BinaryOperator functional interface.
19. Write a program to demonstrate how lambda expressions can simplify anonymous class usage.
20. Implement a program to create a thread pool and execute tasks using lambda expressions.
21. Write a program to demonstrate the use of lambda expressions in a stream pipeline for data processing.
22. Create a program to use lambda expressions to calculate the factorial of a number.
23. Write a program to implement lambda expressions for file filtering based on extensions.
24. Implement a program to demonstrate the use of lambda expressions with Optional class.
25. Write a program to use lambda expressions for grouping elements in a collection.
26. Create a program to demonstrate the use of lambda expressions with the UnaryOperator functional interface.
27. Write a program to use lambda expressions to generate a sequence of numbers.
28. Implement a program to demonstrate the use of method references with lambda expressions.
29. Write a program to use lambda expressions for batch processing of tasks.
30. Create a program to implement a custom stream pipeline using lambda expressions.

Day 23: Exceptions and Handling (30 Questions)

1. Write a program to demonstrate the use of try-catch block to handle exceptions.
2. Create a program to implement custom exception handling.

3. Write a program to demonstrate multiple catch blocks for handling different exceptions.
4. Implement a program to demonstrate the use of finally block.
5. Write a program to demonstrate the use of throw keyword for raising an exception.
6. Create a program to demonstrate the use of throws keyword in method declarations.
7. Write a program to handle NullPointerException.
8. Implement a program to handle ArrayIndexOutOfBoundsException.
9. Write a program to handle ArithmeticException.
10. Create a program to demonstrate nested try-catch blocks.
11. Write a program to demonstrate exception chaining.
12. Implement a program to validate user input and throw a custom exception for invalid data.
13. Write a program to demonstrate the use of try-with-resources for handling file operations.
14. Create a program to handle FileNotFoundException.
15. Write a program to handle IOException while reading a file.
16. Implement a program to demonstrate catching and rethrowing exceptions.
17. Write a program to demonstrate the use of assert for debugging and error checking.
18. Create a program to demonstrate handling ClassNotFoundException.
19. Write a program to demonstrate the use of IllegalArgumentException.
20. Implement a program to demonstrate the use of IllegalStateException.
21. Write a program to demonstrate handling NumberFormatException.
22. Create a program to demonstrate the use of StackOverflowError and how to avoid it.
23. Write a program to demonstrate handling OutOfMemoryError.
24. Implement a program to demonstrate logging exceptions using Logger.
25. Write a program to implement retry logic for handling transient exceptions.
26. Create a program to demonstrate the use of a global exception handler.
27. Write a program to demonstrate the use of error codes in exception messages.
28. Implement a program to use a custom exception hierarchy.
29. Write a program to demonstrate exception handling in a multi-threaded environment.
30. Create a program to demonstrate handling InterruptedException in threads.

Day 24: Threads (30 Questions)

1. Write a program to create a thread by extending Thread class.
2. Implement a program to create a thread by implementing Runnable interface.
3. Write a program to demonstrate thread priorities.
4. Create a program to demonstrate thread states and life cycle.
5. Write a program to demonstrate the use of sleep() method in threads.
6. Implement a program to demonstrate thread synchronization using synchronized keyword.
7. Write a program to demonstrate inter-thread communication using wait() and notify().
8. Create a program to demonstrate the use of join() method in threads.
9. Write a program to implement a producer-consumer problem using threads.
10. Implement a program to demonstrate a deadlock scenario and how to avoid it.
11. Write a program to implement a thread pool using ExecutorService.

12. Create a program to demonstrate the use of Callable and Future for threads.
13. Write a program to demonstrate how to stop a thread gracefully.
14. Implement a program to demonstrate the use of ReentrantLock for thread synchronization.
15. Write a program to implement a countdown timer using threads.
16. Create a program to demonstrate the use of ScheduledExecutorService for scheduling tasks.
17. Write a program to demonstrate the use of ThreadLocal variables.
18. Implement a program to demonstrate the use of ForkJoinPool for parallel tasks.
19. Write a program to demonstrate the use of Semaphore for controlling access to a resource.
20. Create a program to implement a cyclic barrier using CyclicBarrier class.
21. Write a program to demonstrate the use of CountDownLatch for thread coordination.
22. Implement a program to demonstrate the use of atomic variables for thread safety.
23. Write a program to demonstrate the use of Phaser for dynamic thread coordination.
24. Create a program to implement a multi-threaded chat application.
25. Write a program to demonstrate the use of BlockingQueue in producer-consumer problem.
26. Implement a program to demonstrate the use of ReadWriteLock for thread synchronization.
27. Write a program to demonstrate the use of thread-safe collections like ConcurrentHashMap.
28. Create a program to simulate a multi-threaded web server.
29. Write a program to implement a parallel file search using threads.
30. Implement a program to demonstrate thread safety issues and solutions using synchronization.