Java Programming

Day-wise Assignments

Day-1 Topics

1. Intro to Java Programming

--- About Java
--- Features of Java
--- Simple Java Program
--- Java Development Kit(JDK)
--- Java RunTime Environment(JRE)
--- Java Virtual Machine(JVM)
--- Java Just in-time compiler(JIT)
--- Java Memory Management

2. Scope and Variables, Keywords, Data Types, Operators, Java Methods

--- Non-access modifiers: static, final, abstract, synchronized, transient, volatile, strictfp, native
--- Access modifiers: public, protected, default, private
--- Java Unicode System
--- Java Methods(Pass-by-Value)
--- Static Methods
--- Methods Overloading

Day-1 Assignment Questions:

1. Write a code to illustrate the various scope of variables(static, local, instance, block, final variables)?

2. Write a program in which, declare all data types like integer, double, float, long integer, character, byte data, and print them.

3. Write a program to find the maximum of two numbers using the ternary operator.

4. Write a program to check whether the given two numbers are equal or not w/o using comparison operator?

5. Write a program that illustrate the execution order of static block and initializer block?

6. Write a program that illustrates the Explicit type casting for all data types?

7. Write a program to check if a number is a power of 2 using bitwise operator?

8. Write a program to find the first set bit of a number?

9. Write a class Employee with attributes empId, name, department and salary and define a parameterized constructor Employee(int empId, String name, String department, double salary) and assign these variables to instance variables accordingly and display them? Learn variable shadowing.

10. Write a program to check whether the object is an instance of a particular class?

Day-2 Topics

3. Control Statements

--- Selection Statements(if, switch)
--- Iterative Statements(for, while, do...while,...)
--- Jump Statements(labelled jump)

4. Arrays and Strings

--- for each Statement
--- Arrays and its supporting methods
--- String and its methods
--- StringBuffer and StringBuilder

Day-2 Assignment Questions: Part-1

1. Write a program that uses an if statement to find the minimum of three numbers.

2. Write a program to do the following patterns using for loop?

a)
```
1       1       R R R R
  1  1          R     R
    1           R R R R
  1  1          R   R
1       1       R     R
```

3. Write a program to do the following patterns using while loop?

a)
```
      1
    2   3
  4   5   6
    7   8
      9
```

b)
```
w
w  w
w  w  w
w  w
w
```

4. Write a program to do the following patterns using do...while loop?

a) Pascal Triangle

b) ZOHO
CORP
ORAT
IONS

5. Define a method to find the sum of even numbers from the series 1, 2, 3, 4, 5,...n using the continue statement.

6. Define a method to convert the decimal number to a binary number?

7. Use if and switch case statements to find the grade of a student.
Percent>=85 && Percent<=100 Grade ‘A’
Percent>=70 Grade ‘B’
Percent>=50 Grade ‘C’
otherwise print ”Fail”

Day-2 Assignment Questions: Part-2 Arrays

1. Write a program that creates an integer array of 10 elements, accepts values of arrays, and find the sum of all odd numbers.

2. Write a program to take in 10 values and print only those numbers which are prime.

3. Write a program which generates 30 terms of Fibonacci no in an array and then prints it.

4. Design a function void print( ) with a single dimensional array x[] and n (as size of the array) as function arguments. The function calculates the sum of only single digit and sum of only double digit elements from the array. Design a main() function to input size of the array ‘len' and single dimensional array of size 'len', and print the required result by invoking the function print( ).

5. Write a program to initialize the following character arrays and print a suitable message after checking the arrays whether the two arrays are identical or not. Make suitable use of Boolean data type. X[] =('m', 'n', 'o', 'p'} and Y[] =('m', 'n', 'o', 'p' }
X[] =('m', 'n', 'o', 'p'} and
Y[] =('m', 'n', 'o', 'p' }

6. Write a program to accept the year of graduation from school as an integer value from the users. Using the Binary search technique on the sorted array of integers given below. Output the message “record exists" if the value input is located in the array, and if not output the message record does not exits”.
{1982, 1987, 1993, 1996, 1999, 2003, 2006, 2007, 2009, 2010, 2016, 2002}

7. Write a program to input and store the weight of ten people. Sort and display them in descending order using the Selection sort technique.

8. Write a program which takes in 10 people's age and gives the number of people who are <18, 18-60, or >60.

9. Write an array which takes in roll no and marks in three subjects for 10 students. Format and Print the roll no, total marks, and average for all students in a table form.

10. Write a menu driven program to do following operation on two dimensional array A of size m x n. You should use user-defined methods which accept 2-D array A, and its size m and n as arguments. The options are:

1 To input elements into a matrix of size m x n
2 To display elements of a matrix of size m x n
3 Sum of all elements of a matrix of size m x n
4 To display the row-wise sum of the matrix of size m x n
5 To display the column-wise sum of the matrix of size m x n
6 To create a transpose of the matrix of size n x m

Day-2 Assignment Questions: Part-3 Strings

1. Given two binary strings a and b, return their sum as a binary string.
Example-1:
Input: a = "11", b = "1"
Output: "100"
Example-2:
Input: a = "1010", b = "1011"
Output: "10101"
Constraints:
1 <= a.length, b.length <= 10^4
a and b consist only of '0' or '1' characters.
Each string does not contain leading zeros except for the zero itself.

****

2. Given an integer columnNumber, return its corresponding column title as it appears in an Excel sheet.
For example:
A -> 1
B -> 2
C -> 3
...
Z -> 26
AA -> 27
AB -> 28
...
Example 1:
Input: columnNumber = 1
Output: "A"
Example 2:
Input: columnNumber = 28
Output: "AB"
Example 3:
Input: columnNumber = 701
Output: "ZY"
Constraints:
1 <= columnNumber <= 2^31 - 1

****

3. Given a string s, reverse only all the vowels in the string and return it. The vowels are 'a', 'e', 'i', 'o', and 'u', and they can appear in both lower and upper cases, more than once.
Example 1:
Input: s = "hello"

Output: "holle"
Example 2:
Input: s = "leetcode"
Output: "leotcede"
Constraints:
1 <= s.length <= 3 * 10^5
s consist of printable ASCII characters.

****

4. You are given two strings s and t. String t is generated by random shuffling string s and then add one more letter at a random position. Return the letter that was added to t.
Example 1:
Input: s = "abcd", t = "abcde"
Output: "e"
Explanation: 'e' is the letter that was added.
Example 2:
Input: s = "", t = "y"
Output: "y"
Constraints:
0 <= s.length <= 1000
t.length == s.length + 1
s and t consist of lowercase English letters.

****

5. Given two strings s and t, return true if s is a sub-sequence of t, or false otherwise.

A sub-sequence of a string is a new string that is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (i.e., "ace" is a sub-sequence of "abcde" while "aec" is not).
Example 1:
Input: s = "abc", t = "ahbgdc"
Output: true
Example 2:
Input: s = "axc", t = "ahbgdc"
Output: false
Constraints:
0 <= s.length <= 100
0 <= t.length <= 10^4
s and t consist only of lowercase English letters.
Follow up: Suppose there are lots of incoming s, say s1, s2, ..., sk where k >= 10^9, and you want to check one by one to see if t has its sub-sequence. In this scenario, how would you change your code?

****

6. Given two non-negative integers, num1 and num2 represented as string, return the sum of num1 and num2 as a string. You must solve the problem without using any built-in library for handling large integers (such as BigInteger). You must also not convert the inputs to integers directly.
Example 1:
Input: num1 = "11", num2 = "123"
Output: "134"
Example 2:
Input: num1 = "456", num2 = "77"
Output: "533"
Example 3:
Input: num1 = "0", num2 = "0"
Output: "0"
Constraints:
1 <= num1.length, num2.length <= 10^4
num1 and num2 consist of only digits.
num1 and num2 don't have any leading zeros except for the zero itself.

****

7. Given a string s, return the number of segments in the string. A segment is defined to be a contiguous sequence of non-space characters.
Example 1:
Input: s = "Hello, my name is John"
Output: 5
Explanation: The five segments are ["Hello,", "my", "name", "is", "John"]
Example 2:
Input: s = "Hello"
Output: 1
Constraints:
0 <= s.length <= 300
s consists of lowercase and uppercase English letters, digits, or one of the following characters "!@#$%^&*()_+-=',.:"
The only space character in s is ' '.

****

8. We define the usage of capitals in a word to be right when one of the following cases holds:
    1. All letters in this word are capitals, like "USA".
    2. All letters in this word are not capitals, like "leetcode".
    3. Only the first letter in this word is capitalized, like "Google".

Given a string word, return true if the usage of capitals in it is right.
Example 1:
Input: word = "USA"
Output: true
Example 2:
Input: word = "FlaG"
Output: false
Constraints:
1 <= word.length <= 100
word consists of lowercase and uppercase English letters.

Day-3 Topics
5. OOPs Concepts in Java
--- Naming Conventions
--- Class and Objects
--- Access Specifiers
--- Constructors
--- Default Constructor
--- Parameterless Constructors
--- Parameterized Constructors
--- Copy Constructor
--- this keyword in Java

Day-3 Assignment Questions:

1. Create a class named 'Student' with a string variable 'name' and an integer variable 'roll_no'. Assign the value of roll_no as '2' and that of name as "John" by creating an object of the class Student.

2. Create a class named Student that has the following attributes:
name (String)
roll_no (int)
phone_no (int)
address (String)
Create a constructor Student(String name, int roll_no, int phone_no, String address) and store and display the details for two students having names "Sam" and "John" respectively.

3. Write a Java program where you define a class named Employee. Inside the class, define fields to hold an employee’s name, employee ID, designation, and salary.

a. First, create a no-argument constructor that prints a message saying the object has been created, and sets default values for all the fields.

b. Write a parameterized constructor that allows you to set values for all the fields when an object is created.

c. Add another constructor — a copy constructor — that takes an existing employee object and creates a new one with the same values.
In the main method, create:
a. One object using the no-argument constructor,
b. One object using the parameterized constructor,
c. And a third object using the copy constructor.
Finally, display the details of all three employees.

4. Design the Vehicle class that stores information about a vehicle such as its vehicle ID, brand name, and price, such that it can internally keep track of how many vehicles have been created so far, without requiring manual counting from outside the class.

5. Write a program that would print the information (name, year of joining, salary, address) of three employees by creating a class named 'Employee'. The output should be as follows:
Name Year of joining Address
Robert 1994 64C- WallsStreat
Sam 2000 68D- WallsStreat
John 1999 26B- WallsStreat
Can use the format method to format the above said output.

6. Design and implement a custom Java class named MyString that mimics the behavior of Java's built-in String class. Your class should store character data internally (e.g., using a char[] or String as input) and should provide the following string methods,
1. int length() – Returns the number of characters.
2. char charAt(int index) – Returns the character at the specified index.
3. boolean equals(MyString other) – Checks if two MyString objects are equal.
4. MyString toUpperCase() – Returns a new string with all characters in uppercase.
5. MyString toLowerCase() – Returns a new string with all characters in lowercase.
6. MyString substring(int start, int end) – Returns a substring from start to end-1.
7. MyString concat(MyString other) – Concatenates another string to the current one.
8. boolean contains(MyString sub) – Checks if a substring exists.
9. int indexOf(char ch) – Returns the index of the first occurrence of the character.
10. MyString replace(char oldChar, char newChar) – Replaces all occurrences of a character.
    * Implement proper constructors, including one that takes a char[] or String as input.
    * Avoid using any built-in String methods to perform the operations internally.
    * Write a main method to demonstrate the working of your MyString class.

Day-4 Topics

6. Inheritance
--- Types
--- Single Inheritance
--- Multi-Level Inheritance
--- Hierarchical Inheritance
--- Hybrid Inheritance
--- Relationships
--- IS-A relationship
--- HAS-A relationship[Association, Aggregation, Composition]
--- Using Class, Java cannot support Multiple Inheritance

Day-4 Assignment Questions:

1. We want to store the information about different vehicles. Create a class named Vehicle with two data members named mileage and price. Create its two subclasses

---Car with data members to store ownership cost, warranty (by years), seating capacity, and fuel type (diesel or gasoline).
---Bike with data members to store the number of cylinders, number of gears, cooling type(air, liquid or oil), wheel type(alloys or spokes) and fuel tank size(in inches)
Make another two subclasses, Audi and Ford of Car, each having a data member to store the model type.
Next, make two subclasses Bajaj and TVS, each having a data member to store the make-type.
Now, store and print the information of an Audi and a Ford car (i.e. model type, ownership cost, warranty, seating capacity, fuel type, mileage and price.) Do the same for a Bajaj and a TVS bike.

2. Construct a base class called twoD contains x and y and methods to read and write the x and y. Create another class called threeD which is derived from twoD and also contains z and write methods to read and write z. Also write a method to find the distance of two threeD Points.

sqrt((x2-x1)^2+(y2-y1)^2+(z2-z1)^2)

In main:

Create one ThreeD object using the default constructor.
Use the setters to set x, y, and z.
Create the second ThreeD object using the constructor with three arguments.
in the TwoD class:
Add a print statement to the TwoD default constructor with a message "TwoD default constructor"
Add a print statement to other TwoD constructor with a message "TwoD constructor with two arguments"

3. A class Point is declared as follows:

```
class Point{
public:
Point(){int=0;int=0;} //default constructor
void setPoint(int,int); //set coordinates
final int getX(){return x;} //get x coordinates
final int getY() return y;} //get y coordinates
void printPoint(); // print (x,y) coordinates
private int x;
private int y;
};
```

Write the implementation of the class Point in the same file. Then, declare a class called Circle that is derived from the class Point. The class Circle has public member functions Circle (constructor), setRadius(), getRadius() and area() and one private data member radius. The class Circle indirectly uses the private members x and y of class Point to store the coordinate of the center of the circle. The class Circle also checks to make sure the radius value is a positive number; otherwise, it is set to the default value 1.

Note: The private members of class Point can only be indirectly accessed by class Circle using public methods of class point. Implement the class Circle and write a driver program to test the class Circle. An example of the output of the driver program is.

```
Enter x: 2
Enter y: 2
Enter radius: 1
Circle center is (2,2)
Radius is 1
Area is 3.14
```

4. Write a program to illustrate that Classes cannot be used for multiple Inheritance.

5. Create a class named Shape with a method that prints "This is a shape". Create another class named Polygon inheriting the Shape class with the same method that prints "Polygon is a shape". Create two other classes named Rectangle and Triangle having the same method which prints "Rectangle is a polygon" and "Triangle is a polygon" respectively. Again, make another class named Square having the same method which prints "Square is a rectangle".

Now, try calling the method by the object of each of these classes.

6. Design a simple inventory system in Java using object-oriented principles that demonstrates the use of static variables and object containment.
Create two classes:
---Store
---Product

Requirements:

The Store class should have:
a. storeName and storeLocation as static variables since they are common to all products in the store.
b. A static method setStoreDetails(String name, String location) to initialize the static variables.
c. A static method displayStoreDetails() to print store details.
d. A list to maintain multiple Product objects (i.e., the store contains many products).
e. A method addProduct(Product product) to add products to the store.
f. A method displayAllProducts() to display details of all products.

The Product class should have:
a. Product ID, name, price, and quantity as instance variables.
b. A constructor to initialize these fields.
c. A method displayProduct() to show product details.

Task: Implement the above classes and demonstrate their use in the main() method by:
a. Setting store details once.
b. Creating multiple product objects.
c. Adding them to the store.
d. Displaying store and product information.

Also Check how many .class files are generated.

7. Design a calculator application using Java Inheritance. Create the following class hierarchy:

--->BasicCalculator (Base class):

Implement all the basic arithmetic methods, such as:
--->add(int a, int b)
--->subtract(int a, int b)
--->multiply(int a, int b)
--->divide(int a, int b)
--->AdvancedCalculator (Inherits from BasicCalculator)

Add 3 to 4 advanced mathematical operations, such as:
--->power(int base, int exponent)
--->modulus(int a, int b)
--->squareRoot(double number)
--->ScientificCalculator (Inherits from AdvancedCalculator)

Add scientific functions, such as:
--->sin(double angle)
--->cos(double angle)
--->log(double value)
--->exp(double value)

* Demonstrate the use of inheritance by creating an object of ScientificCalculator and calling methods from all three levels of the class hierarchy.
* Use appropriate access specifiers and method overrides where required.
* Add a main() method to test all operations.

Day-5 Topics
7. Polymorphism
--- Types
--- Compile-time Polymorphism

--- Runtime Polymorphism
--- Upcasting
--- Downcasting
--- Covariant return type

Day-5 Assignment Questions:

1. Design a Java program to maintain hospital medical records.

Create a base class named MedicalRecord that includes common attributes:
--- recordId, patientName, dateOfVisit, and diagnosis.
--- inputRecordDetails() – to input common record information.
--- override displayRecord() – to display the common record details.

Create a subclass InPatientRecord that extends MedicalRecord and adds:
--- roomNumber, numberOfDaysAdmitted, roomCharges.
--- calculateTotalCharges() – to compute and return total inpatient cost.
--- displayRecord() – to include all details, including total charges.

Create another subclass OutPatientRecord that extends MedicalRecord and adds:
--- doctorName, consultationFee.
--- override displayRecord() – to include all outpatient-specific details.
Include all the necessary classes if it's needed.

2. Can we override private method, constructor, static method, final method? Illustrate with an example.

3. Create a Java application to manage employees in a company. Define a base class Employee with a method calculateSalary(). Then create two subclasses FullTimeEmployee, PartTimeEmployee and ContractEmployee that override calculateSalary() method based on their working type.

4. Design a Java Ticket Booking System using polymorphism where Bus, Train, and Flight tickets share a common method but implement booking differently.

Day-6 Topics
8. Encapsulation and Packages
--- Binding with example
--- Ways of using packages
--- static import

Day-6 Assignment Questions:

1. Illustrate the concept-Encapsulation with the Payment Gateway System.[The class should encapsulate private data members such as transaction ID, payer name, payee name, amount, payment method, and transaction status. Provide appropriate public getter and setter methods to access and modify these details securely. Also include a method to display the transaction summary.]

2. Create a Java application that demonstrates the use of user-defined packages by organising classes based on their functionality:

--- arithmetic: This package should include a class that defines methods to perform the following operations on two numbers:
a. Addition
b. Subtraction
c. Multiplication
d. Division
e. Modulo

--- stringutils: This package should include a class that defines methods for:
a. Concatenating two strings
b. Reversing a string
c. Finding the length of a string

In your main class (outside those packages), import the above packages and demonstrate the usage of all the methods.

3. Design a class Employee with private data members: employee ID, name, designation, department, and monthly salary. Use proper getter and setter methods to access and update these fields. Add a method to calculate and return the annual salary of the employee.

4. Design a travel booking system using two packages:

- travel.booking – includes a class Ticket with ticket ID, destination, and fare.
- travel.user – includes a class User with user details and a method to book a ticket.

Illustrate accessing ticket data from within the user class.

5. Design a class named BankAccount that uses the concept of encapsulation. The class should have the following private data members: account number, account holder name, and balance. Provide public getter and setter methods to access and modify these fields. Also, include a method to deposit and withdrawal of amount ensuring that the balance cannot go negative.

Day-7 Topics
9. Abstraction
--- Abstract class
--- Interface

Day-7 Assignment Questions:

1. All the banks operating in India are controlled by the RBI. RBI has set a well defined guideline (e.g. minimum interest rate, minimum balance allowed, maximum withdrawal limit etc) which all banks must follow. For example, suppose RBI has set the minimum interest rate applicable to a saving bank account to be 4% annually; however, banks are free to use the 4% interest rate or to set any rates above it.

Write a program to implement bank functionality in the above scenario. Note: Create few classes namely Customer, Account, RBI (Base Class) and few derived classes (SBI, ICICI, PNB etc). Assume and implement required member variables and methods in each class.
Hint:

```
class Customer
{
//Personal Details ...
// Few functions ...
}
class Account
{
// Account Detail ...
// Few functions ...
}
abstract class RBI
{
Customer c; //hasA relationship
Account a; //hasA relationship
..
public abstract double getInterestRate();
public abstract double getWithdrawalLimit();
}
class SBI extends RBI
{
//Use RBI functionality or define own functionality.
}
class ICICI extends RBI
{
//Use RBI functionality or define own functionality.
}
```

2. Design a Payment Gateway System using an interface named PaymentMethod with a method pay(double amount). Implement this interface in different classes like CreditCardPayment, DebitCardPayment, and UPIPayment. Write a main class where you can accept payment using different methods.

3. Create a Java application to manage employees in a company. Define an abstract class Employee with a method calculateSalary(). Then create two subclasses FullTimeEmployee and PartTimeEmployee that override calculateSalary() method based on their working type. Demonstrate runtime polymorphism by calling calculateSalary() on different types of employees using the Employee reference.

4. Create a Java application to manage employees in a company. Define an Interface Employee with a method calculateSalary(). Then create two subclasses FullTimeEmployee and PartTimeEmployee that override calculateSalary() method based on their working type. Demonstrate runtime polymorphism by calling calculateSalary() on different types of employees using the Employee reference.

5. Develop a Java application for a Ticket Booking System that allows users to book tickets for different types of transportation modes such as **Bus**, **Train**, and **Flight**.

- Define a common interface or abstract class Ticket with a method bookTicket() that each transportation mode must implement differently.
- Create classes BusTicket, TrainTicket, and FlightTicket that extend the abstract class or implement the interface.
- Demonstrate **runtime polymorphism** by calling the bookTicket() method using a reference of the base class/interface.

Day-8 Topics
10. Miscellaneous Concepts(Self-Study topics)
--- Object class
--- Object cloning
--- Wrapper class
--- Call-by-Value
--- Recursion
--- Math class
--- Command-Line Arguments
--- Singleton class
--- Comparator and Comparable Interfaces

Day-8 Assignment Questions:

1. Illustrates with an example of using Singleton class.

2. Develop a Java program that illustrates the usage of the Comparator Interface.

3. Develop a Java program which illustrates the usage of Comparable Interface.

4. Get some strings through the command-line prompt and use an array to store and display them.

5. Write a program to illustrate the usage of clone() and find out what kind of copy it will make.

6. Develop a Java program to illustrate pass-by-value.

7. Develop a Java program to illustrate the usage of toString() method.

8. Write a Java program to demonstrate the concept of object cloning using the clone() method.
--->Create a class Student with fields like name, rollNo, and department.
--->Attempt to clone an object of this class using the clone() method.
--->Catch and handle the CloneNotSupportedException if thrown.

***Also find out and explain why the class must implement the Cloneable interface to avoid CloneNotSupportedException.
-->Your program should clearly illustrate:
a. What happens if Cloneable is not implemented
b. How the error is resolved by implementing Cloneable

Day-9 Topics

11. Exception Handling

--- Errors and Exception
--- Types
--- Checked Exception
--- Unchecked Exception
--- try-catch-finally construct
--- throw and throws, the keywords
--- Multiple catches
--- Exception Propagation
--- Exception handling with Inheritance
--- Custom Exception

Day-9 Assignment Questions:

1. int[] arr = {2, 5, 1, 4, 0, 7};
int quotient = arr[7] / arr[4];
Develop a Java program which handles any unexpected situations that may arise during execution.

2. Demonstrate multiple catch blocks: accept two numbers as strings, then convert them to integers, and perform division, and catch the following exceptions: InputMismatchException, NumberFormatException, ArithmeticException and Exception.

3. Write a program to illustrate how to throw a ClassNotFoundException.

4. Write a method to parse a string to an integer. Throw an exception if the string is not a valid number. Handle it using try-catch.

5. Create a program where the try block contains a return statement. Ensure that the finally block executes before the method returns. Show this with output.

6. Write a Java program to accept a 4-digit ATM PIN from the user and validate whether it meets the following conditions:
--->It must be exactly 4 digits long.
--->It should contain only numeric characters.
--->It must not start with 0.
Display an appropriate message whether the PIN is valid or invalid.

7. Write a Java program that shows exception propagation across multiple methods (method1 calls method2 calls method3, which throws the exception). Handle the exception in method1.

8. Design a login system that throws AuthenticationException if the username or password is incorrect. Handle it and display a login failure message.

9. Create a method to read a file from disk. Handle FileNotFoundException and IOException using try-catch-finally.

10. Write a Java program to manage a voting system where a person must be at least 18 years old to be eligible to vote. Use a custom exception to handle the scenario when an ineligible person tries to register for voting. Display appropriate messages for eligible and ineligible voters.

Day-10 Topics

12. Collections

--- Interfaces and Classes
--- Iterable and Iterator Interfaces
--- List, Queue, Set, Map
--- ArrayList, Stack, LinkedList, ArrayDeque, PriorityQueue,
--- HashSet, LinkedHashSet, TreeSet, HashMap, LinkedHashMap, TreeMap

Day-10 Assignment Questions:

1. Write a Java program

a. to create a new array list, add some colors (string) and print out the collection.
b. to iterate through all elements in an array list.
c. to insert an element into the array list at the first position.
d. to retrieve an element (at a specified index) from a given array list.
e. to update specific array element by given element.
f. to remove the third element from an array list.
g. to search an element in an array list.
h. to sort a given array list.
i. to copy one array list into another.
j. to shuffle elements in an array list.

2. Write a Java program to,
a. append the specified element to the end of a linked list.
b. iterate through all elements in a linked list.
c. iterate through all elements in a linked list starting at the specified position.
d. iterate a linked list in reverse order.
e. insert the specified element at the specified position in the linked list.
f. insert elements into the linked list at the first and last position.
g. insert the specified element at the front of a linked list.
h. insert the specified element at the end of a linked list.
i. insert some elements at the specified position into a linked list.
j. get the first and last occurrence of the specified elements in a linked list.

3. Write a Java program to,
a. append the specified element to the end of a hash set.
b. iterate through all elements in a hash list.
c. get the number of elements in a hash set.
d. empty the hash set.
e. test a hash set is empty or not.
f. clone a hash set to another hash set.
g. convert a hash set to an array.
h. convert a hash set to a tree set.
i. convert a hash set to a List/ArrayList.
j. compare two hash set.

4. Write a Java program to,
a. create a new tree set, add some colors (string) and print out the tree set.
b. iterate through all elements in a tree set.
c. add all the elements of a specified tree set to another tree set.
d. create a reverse order view of the elements contained in a given tree set.
e. get the first and last elements in a tree set.
f. clone a tree set list to another tree set.
g. get the number of elements in a tree set.
h. compare two tree sets.
i. Create a TreeSet that stores a set of numbers, find the numbers less than 7 in a tree set.
j. get the element in a tree set which is greater than or equal to the given element.
k. get the element in a tree set which is less than or equal to the given element.
l. get the element in a tree set which is strictly greater than or equal to the given element.
m. get an element in a tree set which is strictly less than the given element.
n. retrieve and remove the first element of a tree set.
o. retrieve and remove the last element of a tree set.
p. remove a given element from a tree set.

5. Write a Java program to,
1. create a new priority queue, add some colors (string) and print out the elements of the priority queue.
2. iterate through all elements in the priority queue.
3. add all the elements of a priority queue to another priority queue.
4. insert a given element into a priority queue.
5. remove all the elements from a priority queue.
6. count the number of elements in a priority queue.
7. compare two priority queues.
8. retrieve the first element of the priority queue.

9. retrieve and remove the first element.
10. convert a priority queue to an array containing all of the elements of the queue.

6. Write a Java program to,
1. associate the specified value with the specified key in a Tree Map.
2. copy a Tree Map content to another Tree Map.
3. search a key in a Tree Map.
4. search a value in a Tree Map.
5. get all keys from the given a Tree Map.
6. delete all elements from a given Tree Map.
7. sort keys in Tree Map by using comparator.
8. get a key-value mapping associated with the greatest key and the least key in a map.
9. get the first (lowest) key and the last (highest) key currently in a map.
10. get a reverse order view of the keys contained in a given map.

7. Write a Java program to,
1. associate the specified value with the specified key in a HashMap.
2. count the number of key-value (size) mappings in a map.
3. copy all of the mappings from the specified map to another map.
4. remove all of the mappings from a map.
5. check whether a map contains key-value mappings (empty) or not.
6. get a shallow copy of a HashMap instance.
7. test if a map contains a mapping for the specified key.
8. test if a map contains a mapping for the specified value.
9. create a set view of the mappings contained in a map.
10. get the value of a specified key in a map.

8. Develop a Java program to manage a list of bank accounts using ArrayList.
--->Create an Account class with the following attributes:
a. accountNumber (int)
b. holderName (String)
c. balance (double)

--->Use auto-boxing to store the balance and interest as Double wrapper objects.
--->Use manual boxing to convert a primitive interest rate into a Double object.
--->Unbox the values (both automatic and manual) to calculate and update the new balance.
--->Add at least three Account objects to an ArrayList<Account>.

--->For each account:
a. Display the holder name, account number, original balance
b. Apply 5% interest
c. Show the new balance using primitive values (unboxed).

Day-11 Topics
13. File and I/O
--- Classes: Files, Directories, Console, FileDescriptor
--- I/O Stream and its Types
--- Byte Stream
--- Character Stream
--- I/O Byte Stream Classes
--- I/O Character Stream Classes
--- Serialization(Self-Study topic)

Day-11 Assignment Questions:

1. Write a Java program to read the contents of a text file and display it on the console.

2. Write a Menu driven Java program a) to read content from the user and write it into another file and b) from the file to another file c) to modify the content of a file d) to search for a particular word in a file and display how many times it appears e) read the content(List of Electronic Products) of a .txt file and copy them to the .csv file.

3. Write a Java program that reads a file and prints the number of lines, words, and characters.

4. Write a Java program to append a text read from the user to an existing file without overwriting the original content.

5. Design a Java application to manage a product inventory system using file handling and object serialization.

Create a class Product with the following attributes:
--->int productId
--->String name
--->double price
--->int quantity

a. Ensure the class implements the Serializable interface.
b. Methods:
--->Add a Product: Accept product details from the user and write the object to a file (products.dat).
--->View All Products: Read all product objects from the file and display them.
--->Search Product by ID: Search for a product by productId in the file and display its details if found.
--->Update Product: Locate a product in the file using the ID, update its price or quantity, and save the changes back to the file.
--->Delete Product: Remove a product object from the file by ID (involves rewriting the file).
--->Use proper exception handling and ensure that object serialization and deserialization are handled safely and efficiently.

Day-12 Topics
14. Multi-Threading
--- Two ways of Creating Threads
--- Synchronization
--- Block level
--- Method level
--- Inter-Thread Communication
--- Deadlock(Self-Study topic)
--- Reentrant Monitor(Self-Study topic)

Day-12 Assignment Questions:

1. Write a Java program to demonstrate multi-threading by extending the Thread class.
--->Create a class MyThread that extends Thread.
--->Override the run() method to display the thread name and a message five times with a delay of 500 milliseconds between prints.
--->In the main() method, create two objects of MyThread and start them.
--->Each thread prints its message independently, showing concurrent execution.

2. Write a Java program to create a thread using the Runnable interface.
--->Create a class TaskRunner that implements Runnable.
--->Inside the run() method, print the current thread name and a task-specific message 10 times with a delay of 1000ms.
--->In the main() method, create two Thread objects passing different TaskRunner instances and start both threads.
--->The console should reflect the concurrent execution of both tasks.

3. Write a Java program where one thread prints only even numbers and another prints only odd numbers from 1 to 20. Synchronize the threads so that they print alternately (i.e., 1 2 3 4 ...).

4. Create a Java program with a shared counter. Spawn 3 threads, where each thread increments the counter 1000 times. Use synchronization to avoid race conditions and display the final value.

5. Implement a basic producer-consumer problem using wait() and notify().
--->Producer thread should add items to a shared buffer.
--->Consumer thread should remove items.
Ensure the buffer size is limited to 5 items.. Use Threads to implement the ATM, where you create threads to check the PIN, another thread to perform the cash withdrawal, another one to check the balance amount and print the receipt.
