Feb Batch 2024 Assignments

Please do not write your answers on the page.

Solve only those assignments that have been covered up until this point. Please do not write your answers on this page. Solve only those assignments that have been covered up until this point.

Data Types:

1. Write a Java program to add two strings:
String a = “Hello”;
String b = “Naveen K”

Expected Output :
Hello Naveen K

2. Write a Java program to print the sum of two numbers.
Test Data:
74 + 36
Expected Output:
110

3. Write a Java program to print the division of two numbers.

k = 50/3
Expected Output : 16

4. Write a Java program to compute the specified expressions and print the output. Go to the editor.
Test Data:

- ((25.5 * 3.5 - 3.5 * 3.5) / (40.5 - 4.5))

Expected Output
2.138888888888889

Solution:
double num = ((25.5 * 3.5 - 3.5 * 3.5) / (40.5 - 4.5));
System.***out***.println(num);

5. Try to concat "Hello Selenium" with a character 't'.

Solution:

```
String a = “Hello Selenium”;
char b = ‘t‘;
System.out.println(a+t);
```

6. Create three int variables having values like : 100, 200, 3400. Add them and concatenate and generate this output String :
"Your Total amount is: 3700".

Solution:

```
int a = 100;
int b = 200;
int c = 3400;
System.out.println(“Your Total  amount is : “ +(a+b+c));
```

7. Print the ASCII value of the character 'h'.

Solution:

```
char ch = ‘h’;
System.out.println((byte)ch);
```

8. Write a program to add 3 to the ASCII value of the character 'd' and print the equivalent character.

Solution:

```
char ch = 'd';//99
int res = ch+3;//100
System.out.println((char) res); //g
```

9. Write a program to find the square of the number 3.9.

Solution:

```
double num = 3.9;
System.out.println(num * num);
```

Or using Math class with pow() method.

```
double num = 3.9;
System.out.println(Math.pow(num, 2));
```

Incremental/Decremental Operators:

1) What will be the output of the following program?

```
public class IncrementDecrementQuiz
{
    public static void main(String[] args)
    {
        int i = 11;

        i = i++ + ++i;
```

2) Guess the output of the following program?

```
public class IncrementDecrementQuiz
{
    public static void main(String[] args)
    {
        int a=11, b=22, c;

        c = a + b + a++ + b++ + ++a + ++b;
```

3) What will be the output of the below program?

```
public class IncrementDecrementQuiz
{
    public static void main(String[] args)
    {
        int i=0;

        i = i++ - --i + ++i - i--;

        System.out.println(i);

    }
}
```

4) Is the below program written correctly?

```
public class IncrementDecrementQuiz
{
    public static void main(String[] args)
```

- {
- boolean b = true;
-

```
            ▪      b++;
            ▪
            ▪      System.out.println(b);
    }
}
```

5) What will be the output of the below program?

```
public class IncrementDecrementQuiz
{
    public static void main(String[] args)
    {
        int i=1, j=2, k=3;

        int m = i-- - j-- - k--;

        System.out.println("i="+i);
        System.out.println("j="+j);
        System.out.println("k="+k);
        System.out.println("m="+m);
    }
}
```

6) What will be the output of the following program?

```
public class IncrementDecrementQuiz
{
    public static void main(String[] args)
    {
        int a=1, b=2;
        System.out.println(--b - ++a + ++b - --a);

    }
}
```

7) What will be the value of i, j and k in the below program?

```
public class IncrementDecrementQuiz
{
    public static void main(String[] args)
    {
```

- int i=19, j=29, k=0;
-
- int m = i-- - j-- - k--;
-
- System.out.println("i="+i);
- System.out.println("j="+j);
- System.out.println("k="+k);

```
    }
}
```

8) What is wrong with the below program? Why it is showing compilation error?

```
public class IncrementDecrementQuiz
{
    public static void main(String[] args)
    {
        int i = 11;

        int j = --(i++);
    }
}
```

9) Guess the value of *p* in the below program?

```
public class IncrementDecrementQuiz
{
    public static void main(String[] args)
    {
        int m = 0, n = 0;

        int p = --m * --n * n-- * m--;

        System.out.println(p);
    }
}
```

10) What will be the output of the following program?

```
public class IncrementDecrementQuiz
{
```

- public static void main(String[] args)

```
    {
        int a=1;
        a = a++ + ++a * --a - a--;

        System.out.println(a);
    }
}
```

11) What will be the outcome of the below program?

```
public class IncrementDecrementQuiz
{
    public static void main(String[] args)
    {
        int a = 11++;

        System.out.println(a);
    }
}
```

12) What will be the outcome of the below program?

```
public class JavaIncrementDecrementQuiz
{
    public static void main(String[] args)
    {
        int ch = 'A';

        System.out.println(ch++);
    }
}
```

13) **What will be the outcome of the below program?**

```
public class JavaIncrementDecrementQuiz
{
    public static void main(String[] args)
    {
        char ch = 'A';

        System.out.println(++ch);
    }
}
```

14) **What will be the outcome of the below program?**

```
public class JavaIncrementDecrementQuiz
{
    public static void main(String[] args)
    {
        double d = 1.5, D = 2.5;

        System.out.println(d++ + ++D);//1.5 + 3.5 = 5.0
    }
}
```

Widening and Narrowing:

Q1: **Widening Casting Assignment**: Write a Java program that demonstrates widening casting. Create variables of different data types such as int, float, double, etc. Assign values to them and then cast them to a higher data type. Finally, print out the original and casted values to observe widening casting in action.

Solution:

```
public class WideningCastingDemo {
    public static void main(String[] args) {
        // Define variables of different data types
        int intValue = 10;
        float floatValue = 20.5f;
        double doubleValue = 30.75;

```

Q2: **Narrowing Casting Assignment:** Develop a Java program to illustrate narrowing casting. Declare variables of higher data types like double, float, long, etc., and assign values to them. Then, cast these variables to lower data types like int, short, byte, etc. Print out the original and casted values to observe narrowing casting in action.

Solution:

```java
public class NarrowingCastingDemo {
    public static void main(String[] args) {
        // Define variables of higher data types
        double doubleValue = 45.75;
        float floatValue = 30.5f;
        long longValue = 1000000000L;

```

If-else and Switch-Case:

Conditional Operators:

Q1.a : Find out the greatest number out of three different given numbers:

Input the 1st number: 25

Input the 2nd number: 78

Input the 3rd number: 87

Expected Output :

The greatest: 87

1.b : Find out the greatest number out of four different given numbers:

Input the 1st number: 25

Input the 2nd number: 78

Input the 3rd number: 87

Input the 4th number: 97

Expected Output :

The greatest: 97

2. Write a Java program to test a number is positive or negative.

Test Data

Input number: 35 -- positive number

Input number: -11 -- negative number

3. WAP to check number is odd or even using If - Else
4. WAP to check given alphabet character is Vowel or Consonant using Switch - Case
5. WAP to run your test cases in a specific environment like: QA, Stage, Dev, UAT, Prod using using Switch - Case

6. WAP to book the specific type of car from the Uber app using Switch - Case.
    a. Car Type: Mini, Sedan, SUV, Premium
    b. If user passes wrong car type, print please select the right car type
7. WAP to launch browsers using If-ElseIf and Switch Case.
    a. Browser: Chrome/Firefox/IE/Safari
    b. If user passes wrong browser, print please pass the right browser name
8. WAP to define the interest rate on the basis of Loan type using Switch Case
    a. Loan Type: Car Loan, Housing Loan, Personal Loan, Education Loan
        i. For Housing Loan, if user's salary is less than 35000 USD - print : NOT APPLICABLE FOR Housing Loan

```
public class LoanInterest {
    public static void main(String[] args) {
        String loanType = "Housing Loan";
        double salary = 30000;
        double interestRate = 0.0;

        switch (loanType) {
```

Short cirucuit && and Logical Operator & Assignments:

Q1: Run the code and observe the output. Change the values of **x** and **y** to see how the combined conditions affect the result.

```
 int x = 5;
        int y = 10;

        if (x > 0 && y > 0 && x < y && x * 2 < y) {
            System.out.println("Hi");
        }

```

Q2: Run the code and note the output. Then, change the values of **a** and **b** to all possible combinations of **true** and **false** and observe the results.

```
boolean a = false;
        boolean b = false;

        boolean result = a && b;
        System.out.println(result);

```

Loops Assignments:

1. WAP to print following output:

I am Batman
I am Batman
I am Batman
I am Batman
I am Batman

2. WAP to print following output:
I am Batman 1
I am Batman 2
I am Batman 3
I am Batman 4
I am Batman 5
I am Batman 6
I am Batman 7
I am Batman 8
I am Batman 9

3. WAP to print 10 to 1 using for, while and do-while loop

4. Write a program in Java to print "Hello World" ten times using while loop

5. Write a program in Java to print all the multiplication of 5 from 1 to 100 using while /for/do-while loop

6. Print all odd and even numbers between 1 to 100

7. What will be the output of this program:

```
int i = 1;
```

```java
while(i<=1){
System.out.println("Hi Java");
}
```

8. Print A-Z , a-z, 0-9 with the respective ASCII numbers the console one using while and for loop.

9. Print the following series using for and while loops:

1. **Even Numbers Series:** Write a Java program to print the series of even numbers from 2 to 100.
2. **Reverse Alphabet Series:** Develop a Java program to print the series of lowercase alphabets in reverse order from 'z' to 'a'.
3. **Floating Point Series:** Write a Java program to print the series of floating-point numbers from 1.0 to 10.0.
4. **Multiples of 9 Series:** Develop a Java program to print the series of numbers where each number is a multiple of 9, starting from 0 and ending at 99.

10. Print only vowels (aeiou) using for and while loop. Start the loop from ‘a‘ to ‘z‘.

Solution:

Using for loop:

```java
public class VowelsForLoop {
    public static void main(String[] args) {
        System.out.println("Vowels using for loop:");
        for(char ch = 'a'; ch <= 'z'; ch++) {
            if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                System.out.print(ch + " ");
            }
        }
    }
}
```

Using while loop:

```java
public class VowelsWhileLoop {
    public static void main(String[] args) {
        System.out.println("Vowels using while loop:");
        char ch = 'a';
        while(ch <= 'z') {
            if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                System.out.print(ch + " ");
            }
            ch++;
        }
    }
}
```

11. Print 1 to 10 and break the loop once you find the multiplication of 7 with a message "bye, see you tomorrow".

Solution:

```
int x = 1;
    while (x <= 10) {
      System.out.println(x);
      if (x % 7 == 0) {
        System.out.println("bye, see you tomorrow");
        break;
      }
```

Array Assignments:

Q1: WAP to delete a specific number from the given array.

int p[] = {1,4,5,2,3,22,31, 2};

--Need to remove 22 from the p[] array.

--output should be: [1, 4, 5, 2, 3, 31, 2]

Solution:

```
int p[] = {1,4,5,2,3,22,31, 2};
    int[] a = new int[p.length-1];

    int j = 0;
    for (int i = 0; i < p.length; i++) {
      if (p[i] != 22) {
        a[j] = p[i];
```

Q2: Write a program to create a static Array, having following cricket data:

--name, age, team name, DOB, gender, Strike Rate

--Try to create multiple Object Arrays for different players

--Try to print all the values of each player on the console using for and for each loops

Q3: Print all the elements of this array using for, for each, while and do-while loop:

int p[] = {1,3,4,5,22,56,89,90}

Q4: Print all the elements of this array in reverse order using for, for each, while and do-while loop:
int p[] = {1,3,4,5,22,56,89,90}

Class and Objects:

```java
public class Student {
  String name;
  int age;
}

```

Assignment 1:

- Create two Student objects and initialize their properties.
- Print the details of each student and the total number of students.
- Modify the age of one student and print the updated details.
- Create another Student object and update the total number of students.
- Print the details of the new student and the updated total number of students.

Solution:

```java
public class Student {
    String name;
    int age;

    public static void main(String[] args) {
     int totalStudents = 0;

```

Assignment 2:

- Define a class named **Book** with the following attributes:
  - **String title**
  - **String author**
  - **int pageCount**
  - **int totalBooks** (to keep track of the total number of books)

In the **main** method:

- Create three **Book** objects and initialize their properties with different values.
- Print the details of each book and the total number of books.

- Modify the **pageCount** attribute of one book and print the updated details.
- Create another **Book** object and update the total number of books.
- Print the details of the new book and the updated total number of books.

Assignment 3:

- Define a class named **Employee** with the following attributes:
  - **String name**
  - **int age**
  - **double salary**
  - **double totalSalary** (to keep track of the total salary of all employees)
- In the **main** method:
  Create three **Employee** objects and initialize their properties with different values.
- Print the details of each employee and the total salary of all employees.
- Give a raise to one employee by increasing their salary, and update the total salary accordingly.
- Create another **Employee** object with a different salary and update the total salary of all employees.
- Print the details of the new employee and the updated total salary of all employees.

Assignment 4:

```
public class Car {
    String brand;
    String model;
    int year;

    public static void main(String[] args) {
        Car car1 = new Car();
        car1.brand = "Toyota";
        car1.model = "Camry";
        car1.year = 2020;

        Car car2 = new Car();
        car2.brand = "Honda";
        car2.model = "Accord";
        car2.year = 2019;

        Car car3 = new Car();
        car3.brand = "Ford";
        car3.model = "Fusion";
        car3.year = 2018;

      Car car4 = new Car();
```

```
        car4.brand = "Audi";
        car4.model = "A6";
        car4.year = 2022;

        System.out.println("Original values:");
        System.out.println(car1.brand + " " + car1.model + " " + c
        System.out.println(car2.brand + " " + car2.model + " " + c
        System.out.println(car3.brand + " " + car3.model + " " + c
          System.out.println(car4.brand + " " + car4.model + " " +


        System.out.println("----------");

        // Assignments:
        // 1. Assign car1 to car2 => car2 = car1
        // 2. Assign car2 to car3 => car3 = car2
        // 3. Assign car3 to car4 => car4 = car3
        // 4. Assign car4 to car1 => car1 = car4


        // Print the values of car1, car2, car3 and car4 after eac
    }
}

```

Methods Assignments:

Methods in Java:

1.Write a program to print the **addition/subtraction/division/multiplication** of two numbers by defining your own method

2. Define a method that returns the product of two double numbers.

____

3.Write a program to print the circumference and area of a circle of radius entered by user by defining your own method.

Solution:

```
public static double calculateCircumference(double radius) {
        return 2 * Math.PI * radius;
    }

    public static double calculateArea(double radius) {
        return Math.PI * radius * radius;
    }
```

____

4. Define two methods to print the maximum and the minimum number respectively among three numbers.

____

5. Define a program to find out whether a given number is even or odd - return true/false.

____

6. A person is eligible to vote if his/her age is greater than or equal to 18.
 Define a method to find out if he/she is eligible to vote. - return true/false

____

7. Write a program which will ask the user to enter his/her marks (out of 100). Define a method that will display grades according to the marks entered as below:

Marks	Grade
91-100	AA
81-90	AB
71-80	BB
61-70	BC
51-60	CD
41-50	DD
<=40	Fail

Solution:

```
public static String calculateGrade(int marks) {



        if (marks >= 91 && marks <= 100) {
            return "AA";
        } else if (marks >= 81 && marks <= 90) {
            return "AB";
        } else if (marks >= 71 && marks <= 80) {
            return "BB";
        } else if (marks >= 61 && marks <= 70) {
            return "BC";
        } else if (marks >= 51 && marks <= 60) {
            return "CD";
        } else if (marks >= 41 && marks <= 50) {
            return "DD";
        } else {
            return "Fail";
        }
    }
```

----

8. Write a program to print the factorial of a number by defining a method named 'Factorial'. Factorial of any number n is represented by n! and is equal to 1*2*3*....*(n-1)*n. E.g.-

4! = 1*2*3*4 = 24

3! = 3*2*1 = 6

2! = 2*1 = 2

Also,

1! = 1

0! = 1

Solution:

```
import java.util.Scanner;

public class Factorial {
    public static int factorial(int n) {
        if (n == 0 || n == 1) {
            return 1;
        } else {
            return n * factorial(n - 1);
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        scanner.close();

        int result = factorial(num);
        System.out.println("The factorial of " + num + " is " + re
    }
}

```

=========Assignments for methods in Java with and without return values, with various parameter types including arrays, boolean, int, double, String, and Object arrays===========

1. **Methods without Return Value (void) and without Parameters:**
   - Assignment: Write a method to print "Hello, World!" to the console without taking any input.
   - Signature: **public void printHello()**
2. **Method with Return Value (int) and with Parameters (int, int):**
   - Assignment: Write a method to add two integers and return the result.
   - Signature: **public int add(int num1, int num2)**

3. **Method with Return Value (boolean) and with Parameters (String, String):**
   - Assignment: Write a method to compare two strings and return **true** if they are equal, **false** otherwise.
   - Signature: **public boolean compareStrings(String str1, String str2)**

Solution:

```
public static void main(String[] args) {
        String str1 = "hello";
        String str2 = "hello";

        boolean result = compareStrings(str1, str2);

        System.out.println("Are the strings equal? " + result);
    }

    public static boolean compareStrings(String str1, String str2)
        // Check if both strings are null
        if (str1 == null && str2 == null) {
            return true;
        }
        // Check if one string is null while the other is not
        else if (str1 == null || str2 == null) {
            return false;
        }
        // Compare the strings
        return str1.equals(str2);
    }
```

4. **Method with Return Value (double) and with Parameters (double[]):**
    - Assignment: Write a method to calculate the average of elements in a double array and return the result.
    - Signature: **public double calculateAverage(double[] numbers)**

Solution:

```
public static double calculateAverage(double[] numbers) {

            double sum = 0.0;
            for(double e : numbers) {
                sum = sum + e;
            }

            return sum/numbers.length;
        }

        public static void main(String[] args) {
            double d[] = {1.2,1.3,1.4,1.5};
            double avg = calculateAverage(d);
            System.out.println(avg);
        }
```

5. **Method with Return Value (String) and with Parameters (String[]):**
    - Assignment: Write a method to concatenate all strings in a string array and return the concatenated string.
    - Signature: **public String concatenateStrings(String[] strings)**

Solution:

```
public class Test {

     public static String concatenateStrings(String[] myStr) {

         String finalStr = "";
         for(String e : myStr) {
             finalStr = finalStr + e;
         }
         return finalStr;

    }

    public static void main(String[] args) {
        String str[] = {"Naveen", "Automation", "Labs", "Java", "S
        String s = concatenateStrings(str);
        System.out.println(s);
    }
}
```

6. **Method with Return Value (Object[]) and with Parameters (Object[]):**
   Assignment: Write a method to reverse the order of elements in an Object array and return the reversed array.
   Signature: **public Object[] reverseArray(Object[] array)**

Solution:

```
public static Object[] reverseArray(Object[] array) {

        int length = array.length;
        Object[] reversedArray = new Object[length];

        // Reverse the order of elements
        int count = 0;
        for (int i = length-1; i >=0; i--) {
            reversedArray[count] = array[i];
            count++;
        }

        return reversedArray;
    }

    public static void main(String[] args) {
        Object[] originalArray = {1, 2, 3, 4, 5};
        Object[] reversedArray = reverseArray(originalArray);
        System.out.println(Arrays.toString(reversedArray));

```

7. **Method without Return Value (void) and with Parameters (int, int[]):**
   Assignment: Write a method to print each element of an int array multiplied by a given factor.
   Signature: **public void printArrayWithFactor(int factor, int[] array)**

Solution:

```
public class Test {

    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};
        printArrayWithFactor(2, array);
    }

    public static void printArrayWithFactor(int factor, int[] arra

        for (int num : array) {
            System.out.print((num * factor) + " ");
        }
        System.out.println(); //2 4 6 8 10
    }
}
```

8. **Method without Return Value (void) and with Parameters (String, boolean):**
   Assignment: Write a method to print a message based on a boolean value (**true** for success, **false** for failure).
   Signature: **public void printStatusMessage(String message, boolean isSuccess)**

```
public static void main(String[] args) {
        printStatusMessage("Operation completed successfully", tru
        printStatusMessage("Operation failed", false);
```

```
    }

    public static void printStatusMessage(String message, boolean
        if (isSuccess) {
            System.out.println("Success: " + message);
        } else {
            System.out.println("Failure: " + message);
        }
    }
```

9. **Method without Return Value (void) and with Parameters (String[]): Will cover this with String Manipulation Concept.**
   Assignment: Write a method to print each element of a string array in reverse order.
   Signature: **public void printReverseStringArray(String[] strings)**

Constructor Assignments

Assignment 1:

Create a Java class named "Person" with the following instance variables:

- name (String)
- age (int)
- gender (char)
- height (double)

Create a constructor for the Person class that takes in the name, age, gender, and height as parameters and initializes the instance variables.

Create a main method that creates two instances of the Person class using the constructor and prints out their information.

Questions:

1. What is the purpose of a constructor in Java?
2. How does a constructor differ from a regular method in Java?
3. Can a Java class have multiple constructors? If so, how are they differentiated?
4. What happens if a constructor is not defined in a Java class?

- If you don’t implement any constructor in your class, the Java compiler inserts default constructor into your code on your behalf. You will not see the default constructor in your source code (the .java file) as it is inserted during compilation and present in the bytecode (.class file).

1. Can a constructor call other methods or constructors within the same class?

   Ans: Yes using this keyword

   Example:

```java
public class MyClass {
    private int value;

    // Constructor with parameter calling another constructor
    public MyClass(int value) {
        // Calling another constructor within the same class
        this(); // Calling the no-argument constructor
        setValue(value); // Calling a method to set the value
    }

    // No-argument constructor
    public MyClass() {
        // Default initialization logic
        System.out.println("No-argument constructor called.");
    }

    // Method to set the value
    public void setValue(int value) {
        this.value = value;
    }

    // Method to get the value
    public int getValue() {
        return value;
    }

    public static void main(String[] args) {
        // Creating an instance using the constructor that calls a
        MyClass obj1 = new MyClass(10);
        System.out.println("Value: " + obj1.getValue()); // Output

        // Creating an instance using the no-argument constructor
        MyClass obj2 = new MyClass();
        obj2.setValue(20); // Calling a method to set the value
        System.out.println("Value: " + obj2.getValue()); // Output
    }
}

```

Assignment 2:

Create a Java class named "Rectangle" with the following instance variables:

- length (double)
- width (double)

Create a default constructor for the Rectangle class that sets both the length and width to 0.0.

Create a constructor for the Rectangle class that takes in the length and width as parameters and initializes the instance variables.

Create a method in the Rectangle class named "calculateArea" that returns the area of the rectangle (length * width).

Create a main method that creates two instances of the Rectangle class using both constructors, calculates and prints out their respective areas.

Questions:

1. What is the purpose of a default constructor in Java?
2. How can you differentiate between a default constructor and a constructor that takes in parameters?
3. What is the access level of a constructor in Java?
4. Can a constructor be private? If so, why would you want to make a constructor private?
5. Can a constructor call a method from another class?

   Ans: Yes, just create the object and call the method of that class.

   Example:

```java
// Class with a non-static method to provide initialization data
class Helper {
    public int getInitialValue() {
        return 100;
    }
}

// Class with a constructor that calls a non-static method from an
class MyClass {
    private int value;

    public MyClass() {
        // Creating an instance of Helper to call its method
        Helper helper = new Helper();
        // Calling a non-static method from another class to get i
        this.value = helper.getInitialValue();
    }

    public int getValue() {
        return value;
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating an instance of MyClass
        MyClass obj = new MyClass();
        System.out.println("Initial Value: " + obj.getValue()); //
    }
}

```

Assignment 3:

Create a Java class named "Employee" with the following instance variables:

- id (int)
- name (String)
- salary (double)

Create a constructor for the Employee class that takes in the id, name, and salary as parameters and initializes the instance variables.

Create getter methods for each of the instance variables.

Create a main method that creates an instance of the Employee class using the constructor, prints out the employee's information using the getter methods, and gives the employee a 10% raise using the setter method for the salary instance variable.

Questions:

1. What is the purpose of a getter method in Java?
2. Can a getter method return void?
3. What is the access level of a getter method in Java?
4. What is the purpose of a setter method in Java?
5. Can a setter method return a value other than void?

Solution:

```
public class Employee {
    // Instance variables
    private int id;
    private String name;
    private double salary;

    // Constructor
    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }

    // Getter methods
    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public double getSalary() {
        return salary;
    }

    // Setter method
    public void setSalary(double salary) {
        this.salary = salary;
    }

    public static void main(String[] args) {
        // Creating an instance of the Employee class using the co
        Employee employee = new Employee(101, "John", 50000.00);

        // Printing out the employee's information using the gette
        System.out.println("Employee Information:");
        System.out.println("ID: " + employee.getId());
        System.out.println("Name: " + employee.getName());
        System.out.println("Salary: $" + employee.getSalary());

        // Giving the employee a 10% raise
        double raisePercentage = 0.10;
        double raiseAmount = employee.getSalary() * raisePercentag
        double newSalary = employee.getSalary() + raiseAmount;
        employee.setSalary(newSalary);

        // Printing the updated salary
        System.out.println("After 10% raise, new salary: $" + empl
    }
```

```
}
```

Assignment 4:

Create a Java class named "Car" with the following instance variables:

- make (String)
- model (String)
- year (int)

Create a constructor for the Car class that takes in the make, model, and year as parameters and initializes the instance variables.

Create a default constructor for the Car class that sets the make, model, and year to "Unknown".

Create a main method that creates three instances of the Car class using both constructors and prints out their information.

Questions:

1. What is the purpose of a default constructor in Java?
2. Can a default constructor take in parameters?
3. Can a constructor be overloaded in Java? If so, what does it mean to overload a constructor?
4. Can you create an object of a Java class without calling a constructor?
5. How are instance variables accessed and modified outside of the class they are defined in?

Solution:

```java
public class Car {
    // Instance variables
    private String make;
    private String model;
    private int year;

    // Constructor with parameters
    public Car(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }

    // Default constructor
    public Car() {
        this.make = "Unknown";
        this.model = "Unknown";
        this.year = -1; // Assuming -1 indicates unknown year
    }

    // Getter methods
    public String getMake() {
        return make;
    }

    public String getModel() {
        return model;
    }

    public int getYear() {
        return year;
    }

    public static void main(String[] args) {
        // Creating instances of the Car class using both construc
        Car car1 = new Car("Toyota", "Camry", 2020);
        Car car2 = new Car("Honda", "Accord", 2019);
        Car car3 = new Car();

        // Printing out the information of each car
        System.out.println("Car 1:");
        System.out.println("Make: " + car1.getMake());
        System.out.println("Model: " + car1.getModel());
        System.out.println("Year: " + car1.getYear());
        System.out.println();

        System.out.println("Car 2:");
        System.out.println("Make: " + car2.getMake());
        System.out.println("Model: " + car2.getModel());
        System.out.println("Year: " + car2.getYear());
        System.out.println();

        System.out.println("Car 3:");
        System.out.println("Make: " + car3.getMake());
        System.out.println("Model: " + car3.getModel());
        System.out.println("Year: " + car3.getYear());
    }
}
```

Assignment 5:

Create a Java class named "BankAccount" with the following instance variables:

- accountNumber (String)
- balance (double)

1. Create a constructor for the BankAccount class that takes in the accountNumber and balance as parameters and initializes the instance variables.

Create a method in the BankAccount class named "deposit" that takes in a double value as a parameter and adds it to the balance instance variable.

Create a method in the BankAccount class named "withdraw" that takes in a double value as a parameter and subtracts it from the balance instance variable.

Create a main method that creates an instance of the BankAccount class using the constructor and performs multiple deposits and withdrawals using the deposit and withdraw methods. Print out the updated balance after each transaction.

Questions:

1. What is the purpose of an instance variable in Java?
2. How are instance variables different from local variables in Java?
3. What is the access level of an instance variable in Java?
4. What is the purpose of a method in Java?
5. Can a method call other methods within the same class?

Encapsulation Assignments:

Assignment 1:

Objective: The objective of this assignment is to create a class that uses encapsulation to protect its data and provide getter and setter methods for accessing the data.

Instructions:

1. Create a class called "Person" with the following private attributes: name (String), age (int), and gender (String).
2. Create getter and setter methods for each attribute.
3. Write a method called "printInfo" that prints out the name, age, and gender of the person.
4. Create an instance of the "Person" class and set its attributes using the setter methods.
5. Call the "printInfo" method to verify that the data was set correctly.

Questions:

- What is encapsulation and how does it relate to object-oriented programming?

- Why is it important to use getter and setter methods instead of accessing attributes directly?
- How can encapsulation improve the security and reliability of a program?
- What is the difference between a private attribute and a public attribute?
- How does encapsulation help with code maintainability and scalability?

Assignment 2:
Objective: The objective of this assignment is to create a class that uses encapsulation to hide its implementation details and provide a simple interface for clients.
Instructions:

1. Create a class called "BankAccount" with the following private attributes: accountNumber (String), balance (double), and owner (String).
2. Create getter and setter methods for each attribute.
3. Write a method called "deposit" that takes a double parameter and adds it to the balance.
4. Write a method called "withdraw" that takes a double parameter and subtracts it from the balance.
5. Write a method called "printStatement" that prints out the account number, owner name, and balance.
6. Create an instance of the "BankAccount" class and set its attributes using the setter methods.
7. Call the "deposit" and "withdraw" methods to modify the balance of the account.
8. Call the "printStatement" method to verify that the data was set correctly.

Questions:

1. How can encapsulation be used to hide implementation details from user of a class?
2. What are the benefits of using private attributes in a class?
3. What is the difference between a getter method and a setter method?
4. How can encapsulation improve the readability of code?

Inheritance Assignment:

Question 1 :

1. Hospital Class (Parent Class)

- Define three methods:
  a. **admitPatient()**: to admit a patient to the hospital.
  b. **treatPatient()**: to treat a patient in the hospital.
  c. **dischargePatient()**: to discharge a patient from the hospital.

2. Child Classes: Apollo, Fortis, Max

- Each child class extends the Hospital class.

- Override two methods from the parent class:
  a. **treatPatient()**: Override this method in each child class to provide specific treatment procedures.
  b. **dischargePatient()**: Override this method in each child class to handle discharge procedures specific to each hospital.
- Define individual methods for each child class, such as:
  For Apollo: **performSurgery()**For Fortis: **prescribeMedication()**For Max: **conductTests()**

3. TestHospital Class

- Create a class named **TestHospital** to test inheritance and polymorphism.
- Inside **TestHospital**, instantiate objects of both parent and child classes.
- Demonstrate:
  a. Using child class objects.
  b. Using parent class objects.
  c. Upcasting: Casting a child class object to a parent class reference.

Question 2:

1. Vehicle Class (Parent Class)

- Define a parent class named **Vehicle** with the following attributes and methods:
  - Attributes:
    i. **brand**: representing the brand of the vehicle.
    ii. **model**: representing the model of the vehicle.
    iii. **year**: representing the manufacturing year of the vehicle.
  - Methods:**start()**: to start the vehicle.**accelerate()**: to accelerate the vehicle.**stop()**: to stop the vehicle.

2. Child Classes: Car, Motorcycle, Truck

- Create three child classes named **Car**, **Motorcycle**, and **Truck**, each extending the **Vehicle** class.
- Define specific attributes and methods for each child class:
  - For **Car**:
    - Attribute: **numDoors** (number of doors)
    - Method: **playMusic()** to play music in the car.
  - For **Motorcycle**:
    - Attribute: **engineType** (type of engine)
    - Method: **wheelie()** to perform a wheelie.
  - For **Truck**:Attribute: **cargoCapacity** (capacity of cargo)Method: **loadCargo()** to load cargo onto the truck.

3. TestVehicle Class

- Create a class named **TestVehicle** to test inheritance and polymorphism.

- Inside **TestVehicle**, instantiate objects of both parent and child classes.
- Demonstrate:
  a. Using child class objects.
  b. Using parent class objects.
  c. Upcasting: Casting a child class object to a parent class reference.

Question 3 :

1. Animal Class (Parent Class)

- Define a parent class named **Animal** with the following attributes and methods:
  - Attributes:
    i. **species**: representing the species of the animal.
    ii. **age**: representing the age of the animal.
  - Methods:**eat()**: to simulate the animal eating.**sleep()**: to simulate the animal sleeping.

2. Child Classes: Mammal, Bird, Reptile

- Create three child classes named **Mammal**, **Bird**, and **Reptile**, each extending the **Animal** class.
- Define specific attributes and methods for each child class:
  - For **Mammal**:
    - Attribute: **furColor** (color of fur)
    - Method: **giveBirth()** to simulate a mammal giving birth.
  - For **Bird**:
    - Attribute: **wingSpan** (length of wingspan)
    - Method: **fly()** to simulate a bird flying.
  - For **Reptile**:Attribute: **scalePattern** (pattern of scales)Method: **crawl()** to simulate a reptile crawling.

3. Grandchild Classes: Dog, Eagle, Snake

- Create three grandchild classes named **Dog**, **Eagle**, and **Snake**, each extending one of the child classes (**Mammal**, **Bird**, **Reptile**).
- Define specific attributes and methods for each grandchild class:
  - For **Dog** (extends **Mammal**):
    - Attribute: **breed** (breed of the dog)
    - Method: **bark()** to simulate a dog barking.
  - For **Eagle** (extends **Bird**):
    - Attribute: **nestLocation** (location of the eagle's nest)
    - Method: **hunt()** to simulate an eagle hunting for prey.
  - For **Snake** (extends **Reptile**):Attribute: **venomous** (boolean indicating if the snake is venomous)Method: **strike()** to simulate a snake striking.

4. TestAnimal Class

- Create a class named **TestAnimal** to test inheritance and polymorphism.
- Inside **TestAnimal**, instantiate objects of grandchild classes.
- Demonstrate:
    a. Using grandchild class objects.
    b. Using parent and child class objects.
    c. Upcasting: Casting a grandchild class object to a parent class reference.

String Assignments:

1. Write a program to check two different strings equality.
2. Remove all spaces in a String .

For example : “ Hello Everyone “ . Expected result: “HelloEveryone”.

3. Write a program that will print out the last character and first character of a word.

4. Write a program to verify a word or a character contained in the sentence.

5.Write a function/ method to reverse your own name.

6. Write a program that gives you the last half of the string.

7.Write a program to get the 3rd “ e “ of the string .

For example: “Welcome to Naveen Automation Labs ! “.

8. Write a method which gives an index of (-1) if string is not available. . it should return integer. if String is present, then it should return the specific index.

9. Write a program that breaks a whole string into small strings, and prints out its all values . (Hint: split, loop) .

10. Assume that a string consists of 3 words, print out the middle one.

11. get only numeric part from this String:
String s = "your transaction id is: 12345 and reference id is 34567";