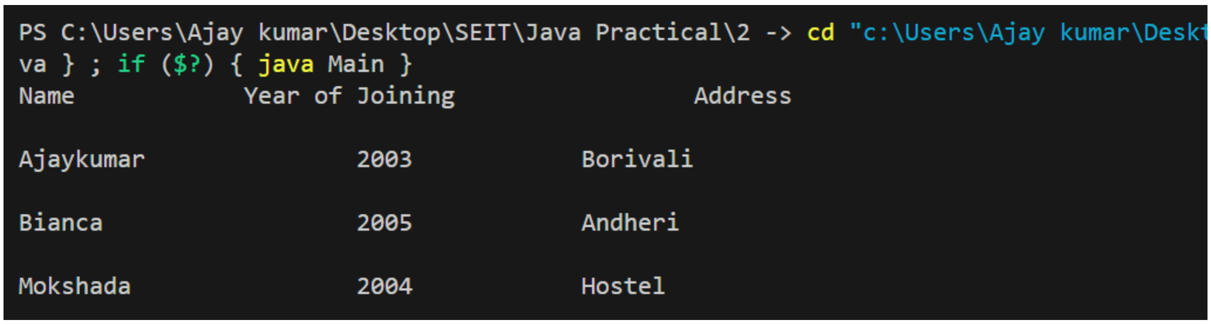
1. Write a program that would print the information (name, year of joining, salary, address) of three employees by creating a class named 'Employee'.

Code:

```
import java.util.*;

class Employee {
    String name, address;
    int yoj;

    public void setDetail (String n, String a, int y) {
        this.name = n;
        this.address = a;
        this.yoj = y;
    }
}
class Main {
    public static void main(String[] args) {
        // Scanner sc = new Scanner();
        Employee emp1 = new Employee();
        emp1.setDetail("Ajaykumar", "Borivali", 2003);
        Employee emp2 = new Employee();
        emp2.setDetail("Bianca   ", "Andheri", 2005);
        Employee emp3 = new Employee();
        emp3.setDetail("Mokshada ", "Hostel ", 2004);

        System.out.println("Name\t\tYear of Joining \t\tAddress\n");
        System.out.println(emp1.name+"\t\t"+emp1.yoj+"\t\t"+emp1.addres
s+"\n");
        System.out.println(emp2.name+"\t\t"+emp2.yoj+"\t\t"+emp2.addres
s+"\n");
        System.out.println(emp3.name+"\t\t"+emp3.yoj+"\t\t"+emp3.addres
s+"\n");

    }
}
```

Output:



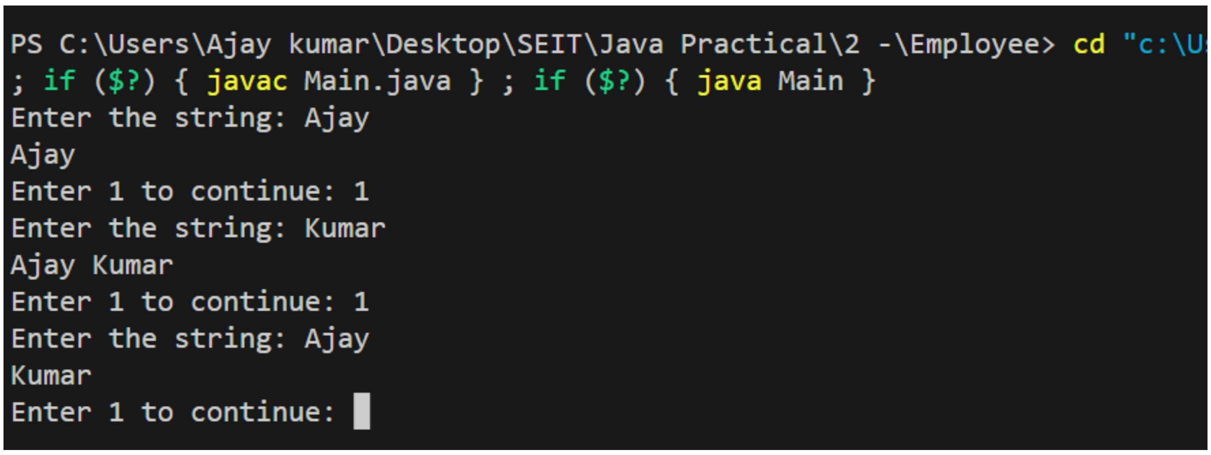
```
PS C:\Users\Ajay kumar\Desktop\SEIT\Java Practical\2 -> cd "c:\Users\Ajay kumar\Deskt
va } ; if ($?) { java Main }
Name            Year of Joining                 Address

Ajaykumar               2003            Borivali

Bianca                  2005            Andheri

Mokshada                2004            Hostel
```

2. Write a java program to add n strings in a vector array. Input new string and check whether it is present in the vector. If it is present delete it otherwise add it to the vector.

Code:

```java
import java.util.*;
import java.lang.*;

public class Main {
    public static void main(String[] arg) {
        Scanner sc = new Scanner(System.in);
        Vector<String> vec = new Vector<String>();
        int on = 1;
        while (on == 1) {
            System.out.println("Enter the string: ");
        String input = sc.next();
            if (vec.indexOf(input) == -1) {
                vec.add(input);
            } else {
                vec.remove(vec.indexOf(input));
            }
            for (int i = 0; i < vec.size(); i++) {
                System.out.println(vec.get(i));
            }
            System.out.println("Enter 1 to continue: ");
            on = sc.nextInt();
        }

        sc.close();
    }
}
```

Output:



```
PS C:\Users\Ajay kumar\Desktop\SEIT\Java Practical\2 -\Employee> cd "c:\U
; if ($?) { javac Main.java } ; if ($?) { java Main }
Enter the string: Ajay
Ajay
Enter 1 to continue: 1
Enter the string: Kumar
Ajay Kumar
Enter 1 to continue: 1
Enter the string: Ajay
Kumar
Enter 1 to continue:
```

1. Explain different array declaration with example

Arrays are used to store multiple values of the same data type in a contiguous memory location. There are different ways to declare arrays in Java, depending on the context and syntax you prefer. Here are the different array declaration methods with examples:

1. **Basic Array Declaration:**

This is the most common way to declare an array in Java.

```
// Declaration with initialization

dataType[] arrayName = {value1, value2, value3, ...};

// Example

int[] numbers = {10, 20, 30, 40, 50};
```

2. **Array Declaration with Size:**

You can also declare an array and specify its size without initializing the values immediately.

```
dataType[] arrayName = new dataType[arraySize];

// Example

double[] prices = new double[5];
```

3. **Array Declaration using the new Keyword:**

This method allows you to declare and initialize an array separately.

```
dataType[] arrayName;

arrayName = new dataType[] {value1, value2, value3, ...};

// Example

String[] fruits;

fruits = new String[] {"Apple", "Banana", "Orange"};
```

2. Differentiate between Array and Vector in java.

Arrays and Vectors are both used for storing collections of elements in Java, but they have some differences in terms of functionality, flexibility, and usage. Here's a comparison between arrays and vectors in Java:

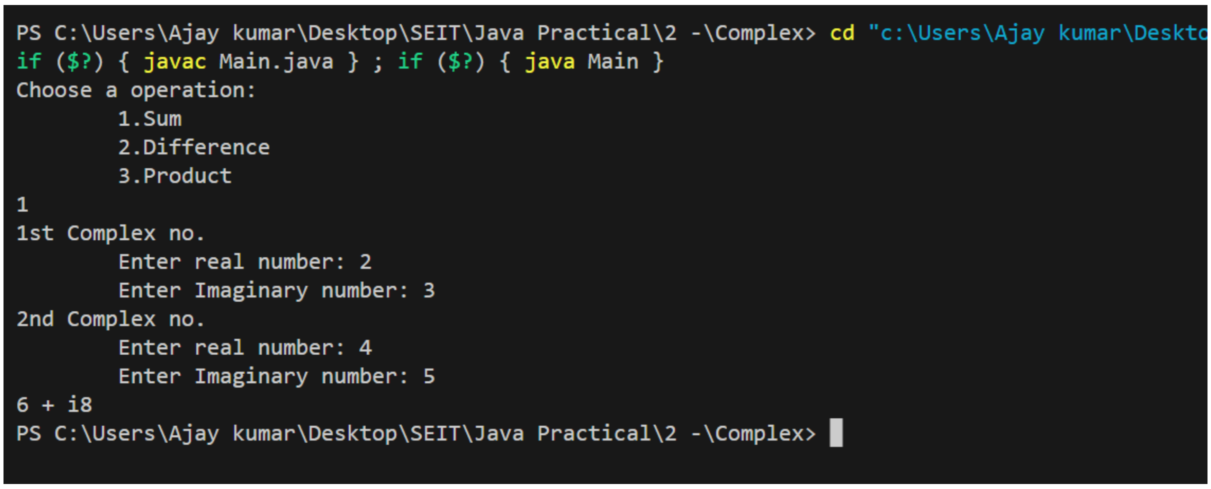
	Array	**Vector**
Mutability	Arrays have a fixed size that is determined at the time of declaration. Once an array is created, its size cannot be changed.	Vectors can dynamically grow or shrink in size as elements are added or removed. Vectors automatically resize themselves
Type Safety	Arrays can store elements of primitive data types as well as objects.	Vectors can only store objects, not primitive data types. They use Java's object boxing to store primitive values.
Synchronization	Arrays are not synchronized, meaning they are not thread-safe. If multiple threads access an array simultaneously and at least one of them modifies it, you need to provide external synchronization.	Vectors are synchronized by default. All methods in the Vector class are synchronized, ensuring thread safety. This can impact performance when compared to unsynchronized collections like ArrayList.
Performance	Arrays are generally faster and have lower memory overhead compared to vectors due to their simplicity.	Vectors may have slightly slower performance due to the synchronization overhead, making them less efficient in cases where synchronization is not required.
Usage	Arrays are often used when you know the size of the collection at the time of creation and don't need to modify the size later.	Vectors are suitable when you need a dynamically resizable collection that is thread-safe. However, due to their synchronization overhead, they might not be the best choice in high-performance applications.
Legacy	Arrays are a fundamental data structure in Java and are present since the beginning.	Vectors are part of the Java Collections framework and were introduced to provide a synchronized version of dynamic arrays. However, they are considered somewhat outdated now, and ArrayList or other modern collections are preferred in most cases.

3. Print the sum, difference and product of two complex numbers by creating a class named 'Complex' with separate methods for each operation whose real and imaginary parts are entered by user.

Code:

```
import java.lang.Math;
import java.util.Scanner;

class Complex {
  int real;
  int imaginary;

  void sum(int x1, int y1, int x2, int y2) {
    System.out.printf("%d + i%d", (x1 + x2), (y1 + y2));
  }

  void difference(int x1, int y1, int x2, int y2) {
    System.out.printf("%d + i%d", Math.abs(x1 - x2), Math.abs(y1 -
y2));
  }

  void product(int x1, int y1, int x2, int y2) {
    System.out.printf("%d + i%d", (x1 * x2 - y1 * y2), (x1 * y2 + x2 *
y1));
  }
}

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    Complex z = new Complex();

    System.out.println("Choose a
operation:\n\t1.Sum\n\t2.Difference\n\t3.Product\n\n");
    int choice = sc.nextInt();

    System.out.printf("1st Complex no.\n\tEnter real number: ");
    int real1 = sc.nextInt();
    System.out.printf("\tEnter Imaginary number: ");
    int imag1 = sc.nextInt();

    System.out.printf("2nd Complex no.\n\tEnter real number: ");
    int real2 = sc.nextInt();
    System.out.printf("\tEnter Imaginary number: ");
    int imag2 = sc.nextInt();
    switch (choice) {
      case 1:
        z.sum(real1, imag1, real2, imag2);
        break;
      case 2:
        z.difference(real1, imag1, real2, imag2);
        break;
      case 3:
        z.product(real1, imag1, real2, imag2);
        break;
      default:
        System.out.println("Error 404: Operation not found");
        break;
    }
    sc.close();
  }
}
```

Output:



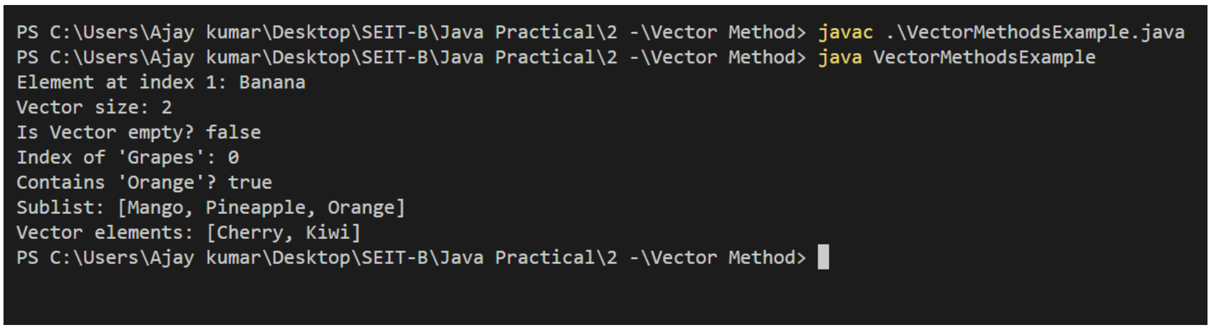
```
PS C:\Users\Ajay kumar\Desktop\SEIT\Java Practical\2 -\Complex> cd "c:\Users\Ajay kumar\Deskto
if ($?) { javac Main.java } ; if ($?) { java Main }
Choose a operation:
        1.Sum
        2.Difference
        3.Product
1
1st Complex no.
        Enter real number: 2
        Enter Imaginary number: 3
2nd Complex no.
        Enter real number: 4
        Enter Imaginary number: 5
6 + i8
PS C:\Users\Ajay kumar\Desktop\SEIT\Java Practical\2 -\Complex>
```

1. Write a Java program to implement 15 methods of Vector class.

Code:

```java
import java.util.Vector;

public class VectorMethodsExample {
    public static void main(String[] args) {
        // Creating a Vector
        Vector<String> vector = new Vector<>();

        // Adding elements to the Vector
        vector.add("Apple");
        vector.add("Banana");
        vector.add("Orange");

        // Accessing elements
        System.out.println("Element at index 1: " + vector.get(1));

        // Changing an element
        vector.set(1, "Grapes");

        // Removing an element
        vector.remove(0);

        // Size of the Vector
        System.out.println("Vector size: " + vector.size());

        // Checking if the Vector is empty
        System.out.println("Is Vector empty? " + vector.isEmpty());

        // Index of an element
        System.out.println("Index of 'Grapes': " +
vector.indexOf("Grapes"));

        // Checking if an element exists
        System.out.println("Contains 'Orange'? " +
vector.contains("Orange"));

        // Adding elements at specific index
        vector.add(1, "Mango");
        vector.add(2, "Pineapple");

        // Sublist
        System.out.println("Sublist: " + vector.subList(1, 4));

        // Clearing the Vector
        vector.clear();

        // Adding all elements from another collection
        Vector<String> newElements = new Vector<>();
        newElements.add("Cherry");
        newElements.add("Kiwi");
        vector.addAll(newElements);

        // Converting Vector to an Array
        String[] array = vector.toArray(new String[0]);

        // Printing elements
        System.out.println("Vector elements: " + vector);
    }
}
```

Output:



```
PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\2 -\Vector Method> javac .\VectorMethodsExample.java
PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\2 -\Vector Method> java VectorMethodsExample
Element at index 1: Banana
Vector size: 2
Is Vector empty? false
Index of 'Grapes': 0
Contains 'Orange'? true
Sublist: [Mango, Pineapple, Orange]
Vector elements: [Cherry, Kiwi]
PS C:\Users\Ajay kumar\Desktop\SEIT-B\Java Practical\2 -\Vector Method>
```