AIM:

Create an object for eac of the two classes and print the percentage of marks for both the students

Program:

```
abstract class Marks {     abstract double
getPercentage();
}

// Define the 'A' class for student A class A
extends Marks {     private int subject1;     private
int subject2;     private int subject3;

    public A(int subject1, int subject2, int subject3) {
this.subject1 = subject1;         this.subject2 = subject2;
this.subject3 = subject3;
    }

    @Override     double getPercentage() {
        int totalMarks = subject1 + subject2 + subject3;         return (double)
totalMarks / (3 * 100) * 100;

    }
}

// Define the 'B' class for student B class B
extends Marks {     private int subject1;     private
int subject2;     private int subject3;     private int
subject4;
```

```
    public B(int subject1, int subject2, int subject3, int subject4) {
this.subject1 = subject1;        this.subject2 = subject2;        this.subject3 =
subject3;        this.subject4 = subject4;
    }

    @Override     double getPercentage() {
        int totalMarks = subject1 + subject2 + subject3 + subject4;        return (double)
totalMarks / (4 * 100) * 100;
    }
}

public class Main {
    public static void main(String[] args) {
        // Create objects for both students
A studentA = new A(85, 90, 78);
B studentB = new B(92, 88, 75, 96);

        // Calculate and print the percentage of marks for both students
        System.out.printf("Student A's Percentage: %.2f%%\n",
studentA.getPercentage());
        System.out.printf("Student B's Percentage: %.2f%%\n",
studentB.getPercentage());
    }
}
```

Algorithm:

Step 1: Create an abstract class called **Marks** with an abstract method **getPercentage()**.

Step 2: Create a class **A** that inherits from **Marks**:

Add instance variables for the marks in three subjects.

- Create a constructor for **A** that takes three subject marks as parameters.
- Implement the **getPercentage()** method in class **A** to calculate the percentage based on the marks in three subjects.

Step 3: Create a class **B** that inherits from **Marks**:

- Add instance variables for the marks in four subjects.
- Create a constructor for **B** that takes four subject marks as parameters.
- Implement the **getPercentage()** method in class **B** to calculate the percentage based on the marks in four subjects.

Step 4:

- Create an object of class **A** with marks for student A.
- Create an object of class **B** with marks for student B.
- Call the **getPercentage()** method on both objects to calculate the percentages.
- Print the percentages obtained by both students

Step 5: End

AIM:

The code to perform operations on a queue object

Program:

```
interface QueueOperations {    void
enqueue(int item);    int dequeue();
boolean isEmpty();       boolean isFull();
}

class MyQueue implements QueueOperations {    private
int[] queue;     private int front;       private int rear;
private int maxSize;
```

```
    public MyQueue(int size) {
        maxSize = size;        queue = new
int[maxSize];        front = -1;        rear = -1;
    }


    public void enqueue(int item) {
        if (isFull()) {
            System.out.println("Queue is full. Cannot enqueue.");
        } else {            if (isEmpty()) {
                front = 0; // If the queue is empty, set front to 0
            }
            rear++;            queue[rear] =
item;
            System.out.println("Enqueued: " + item);
        }
    }

    public int dequeue() {        if
(isEmpty()) {
            System.out.println("Queue is empty. Cannot dequeue.");            return -1; //
Return -1 to indicate an empty queue
        } else {
            int item = queue[front];
            if (front == rear) {
front = -1;                rear = -1;            }
else {                front++;
            }
            System.out.println("Dequeued: " + item);            return item;
        }
    }

    public boolean isEmpty() {        return (front == -1
&& rear == -1);
    }

    public boolean isFull() {        return (rear ==
maxSize - 1);
    }
}

public class Main {
    public static void main(String[] args) {
        MyQueue queue = new MyQueue(5);

        queue.enqueue(10);        queue.enqueue(20);
queue.enqueue(30);



        queue.dequeue();        queue.dequeue();

        queue.enqueue(40);        queue.enqueue(50);



        queue.dequeue();        queue.dequeue();
        queue.dequeue(); // Trying to dequeue from an empty queue



        System.out.println("Is the queue empty? " + queue.isEmpty());
    }
}
```

Algorithm:

Step 1: Create an integer array to store the queue elements.

Step 2: Maintain two integer variables: **front** and **rear** to keep track of the front and rear positions of the queue.

Step 3: Initialize the queue size in the constructor and initialize **front** and **rear** to -1 to indicate an empty queue.

Step 4: Implement the **enqueue** method to add items to the rear of the queue.

Step 5: Implement the **dequeue** method to remove and return items from the front of the queue.

Step 6: Implement the **isEmpty** method to check if the queue is empty.

Step 7: Implement the **isFull** method to check if the queue is full.

Step 8: End

Result:

The above programe are successfully executed in Java