

Lesson 05 Demo 04

Implementing PriorityQueue

Objective: To demonstrate the use of priority queue in Java for sorting and processing data with a first-in-first-out approach

Tools Required: Eclipse IDE

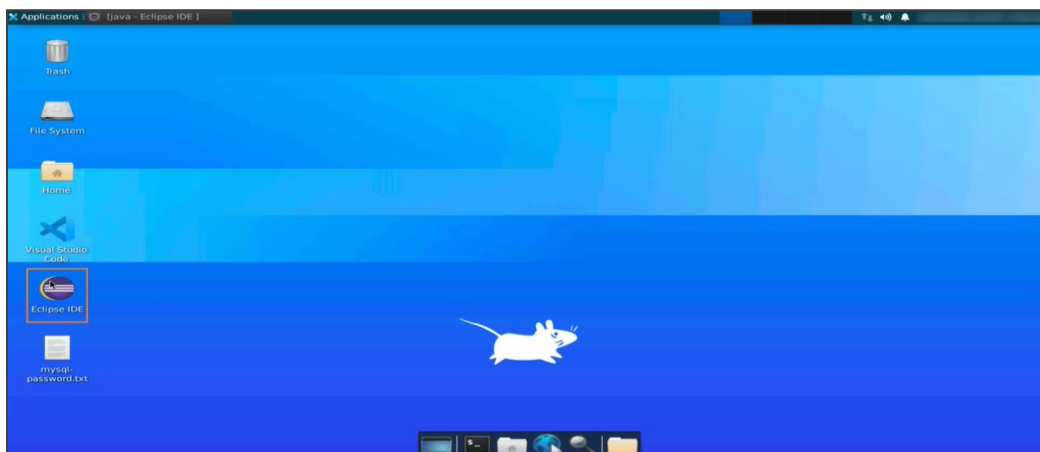
Prerequisites: None

Steps to be followed:

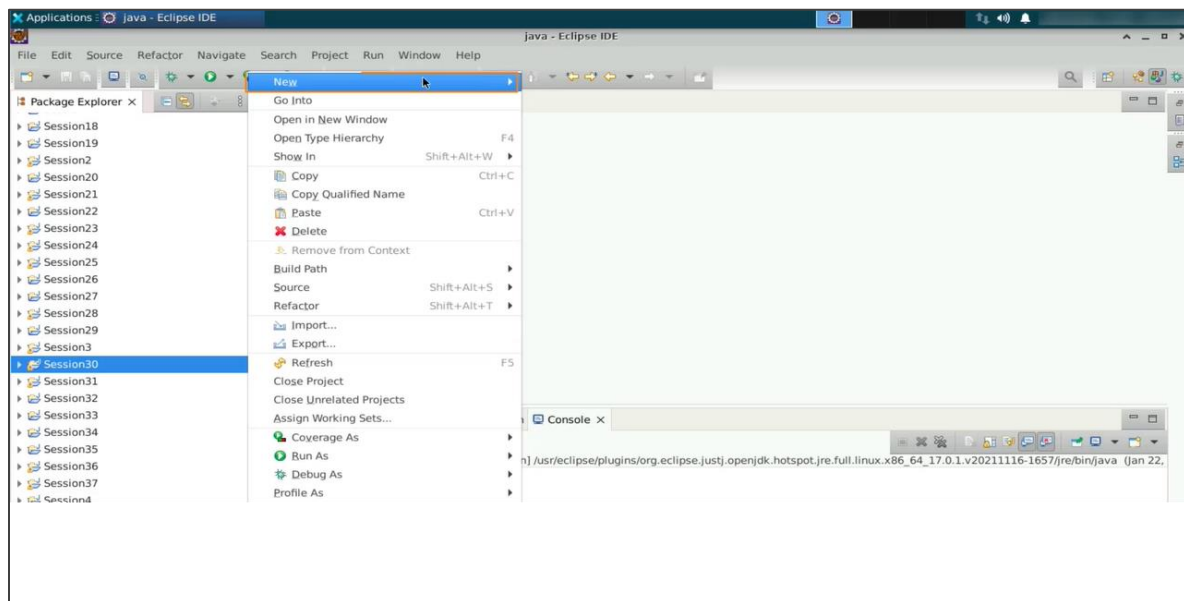
1. Create a new project
2. Add data to the queue using the add method
3. Implement the use of peek and poll methods
4. Implement the use of the variable size using the **queue.size()** method
5. Use the **queue.iterator()** function with example data
6. Implement queue prioritization

Step 1: Create a new project

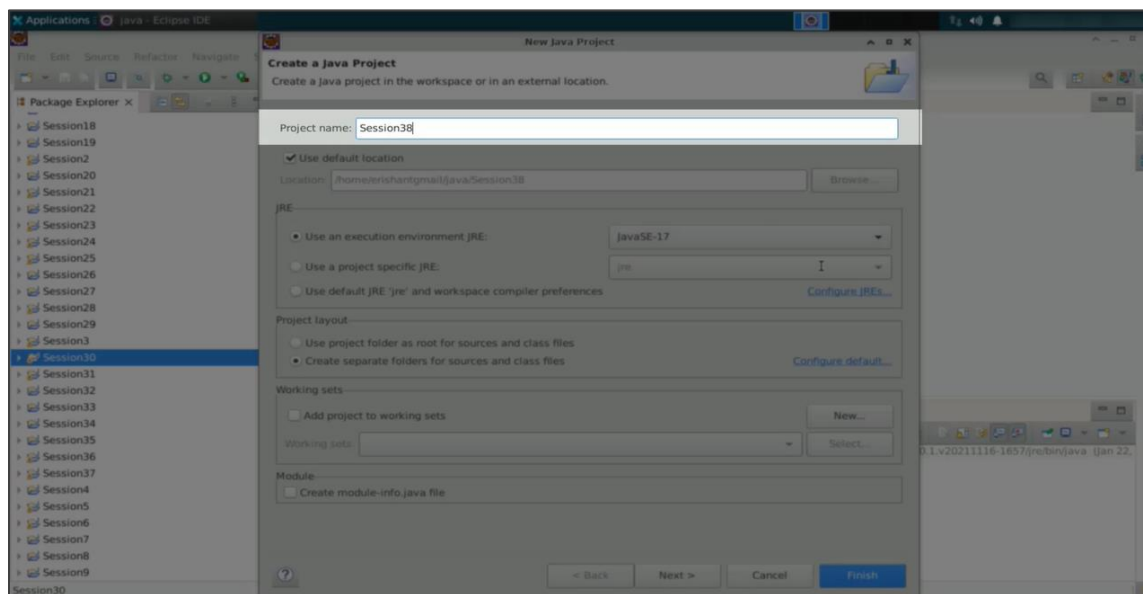
1.1 To create a queue, open the **Eclipse IDE**



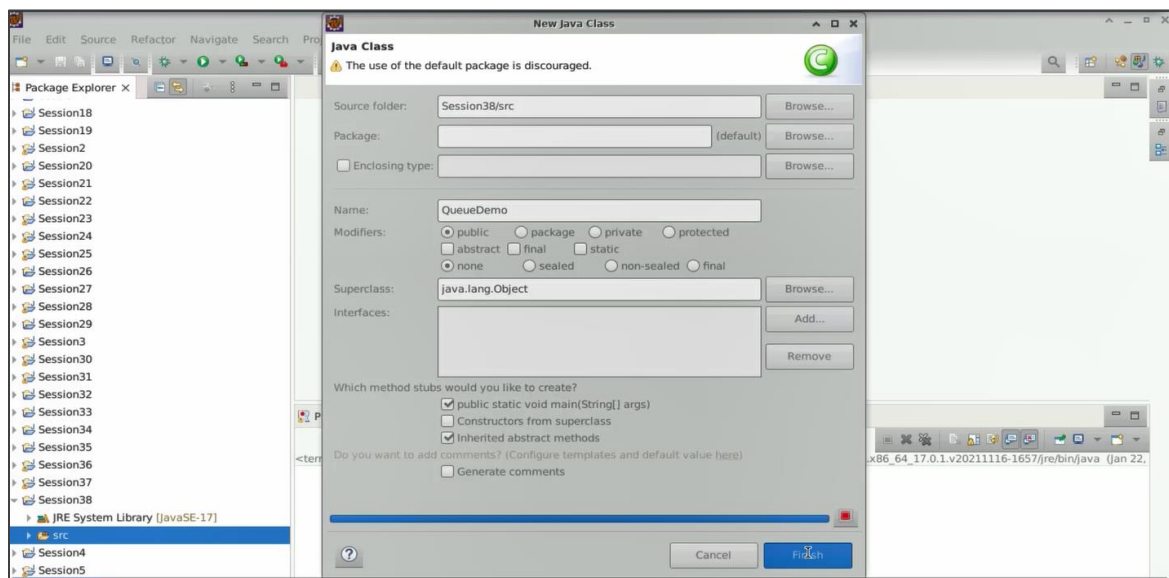
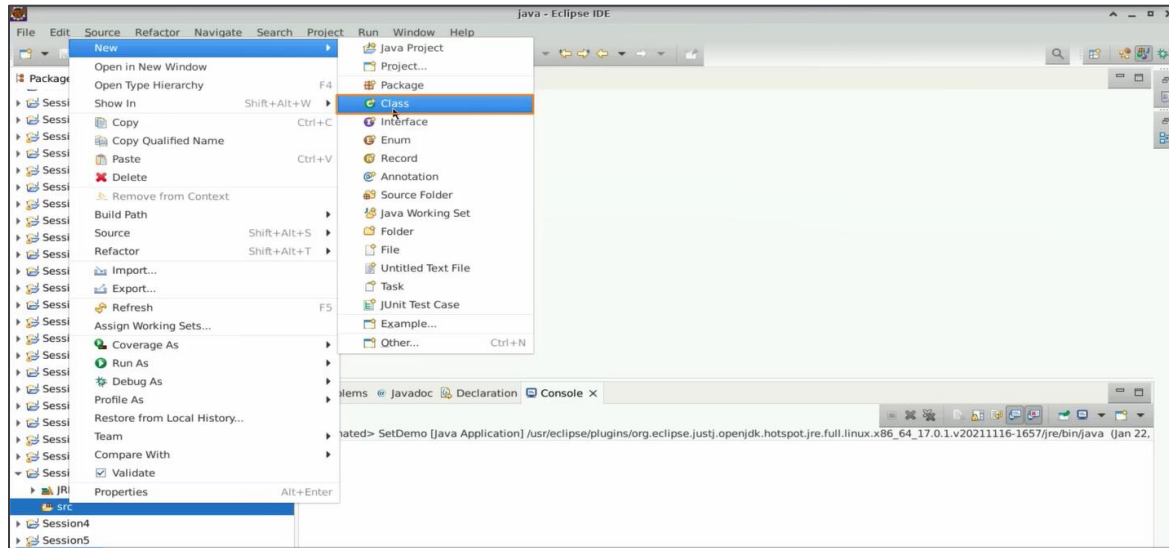
1.2 Create a new Java project



1.3 Name the new project as **Session38**

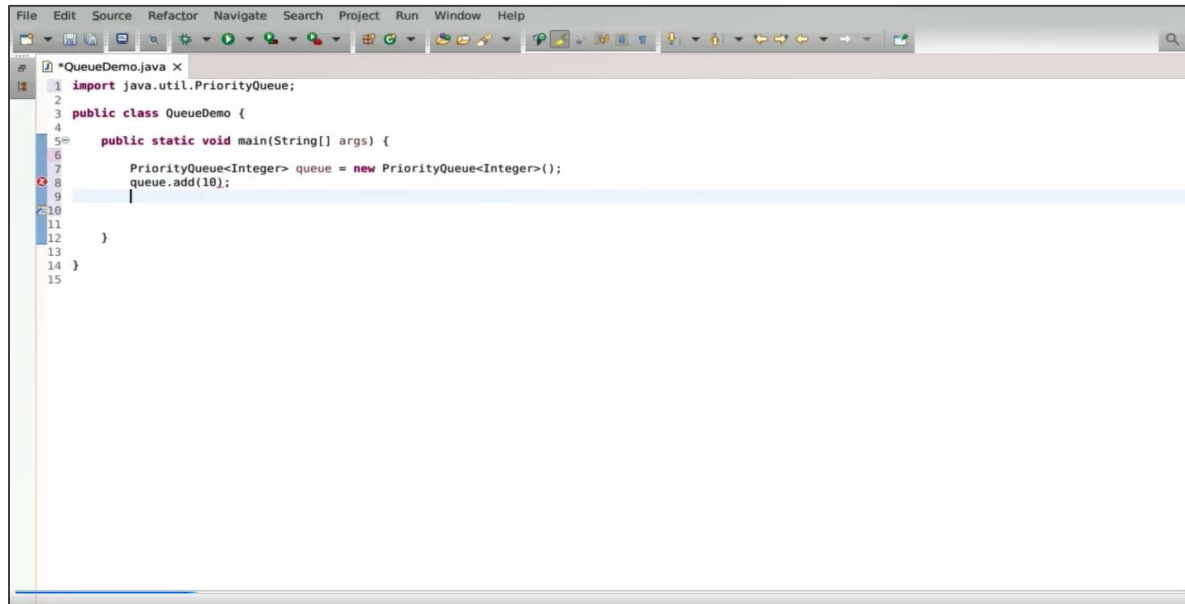


1.4 Right-click on **Session38** and create a new class called **QueueDemo** with the main method



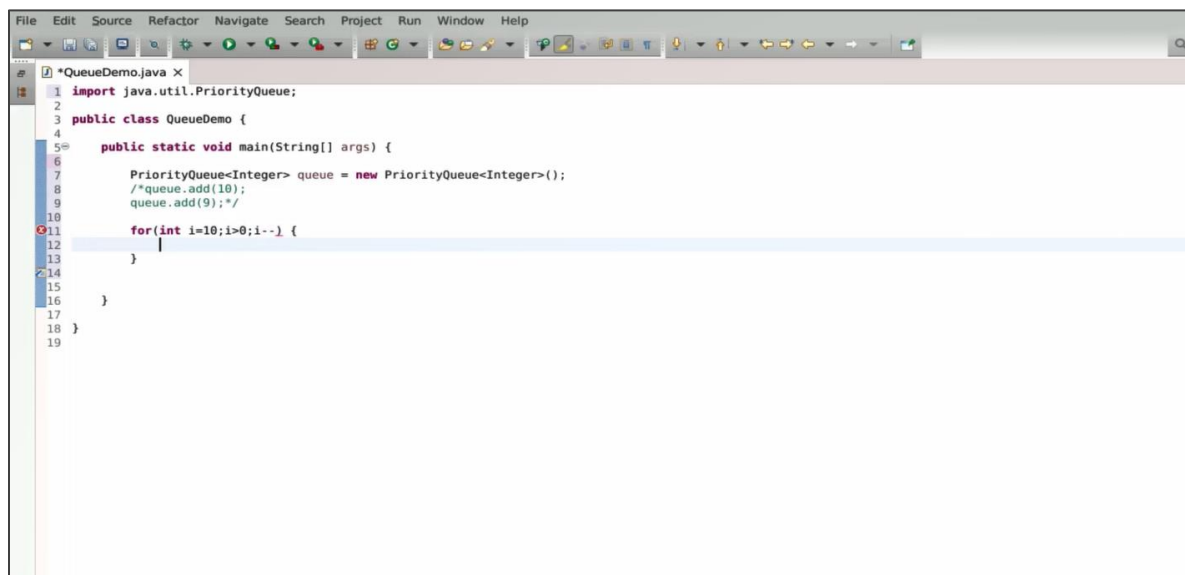
Step 2: Add data to the queue using the add method

2.1 To add data to the queue, use the add method like lists or sets. Write: **queue.add(10);**



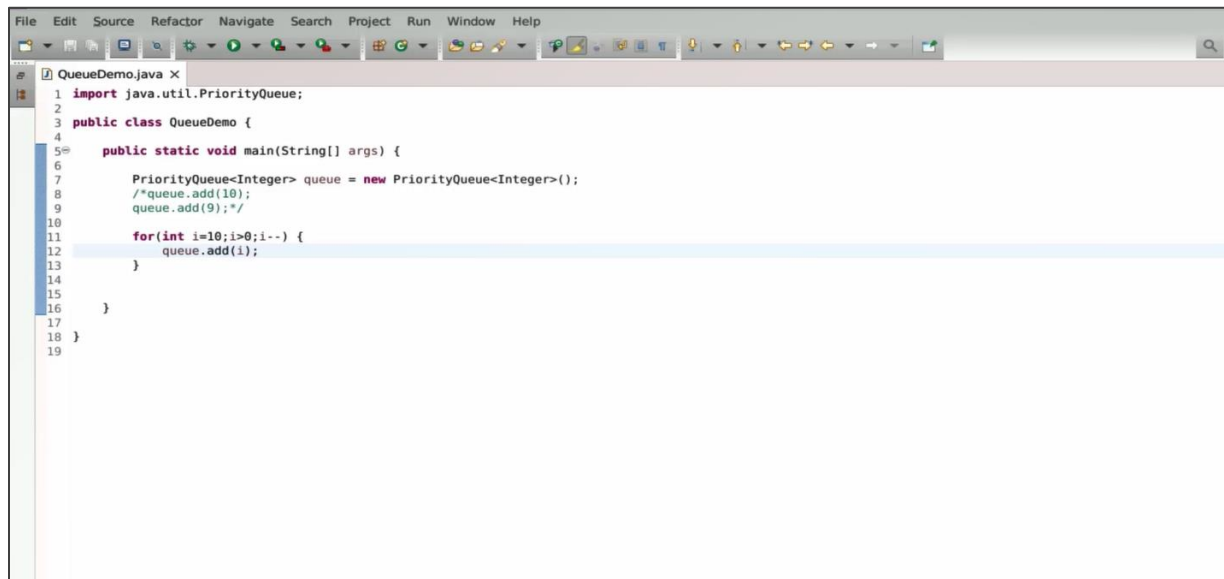
```
1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         queue.add(10);
9
10    }
11
12 }
13
14 }
15 }
```

2.2 Add more data to the queue, such as 9, or alternatively use a loop to add 10 more numbers



```
1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         queue.add(10);
9         queue.add(9);
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15     }
16 }
17
18 }
19 }
```

2.3 Use a reverse loop starting from 10 and decrementing i. Write: **queue.add(i);**



```

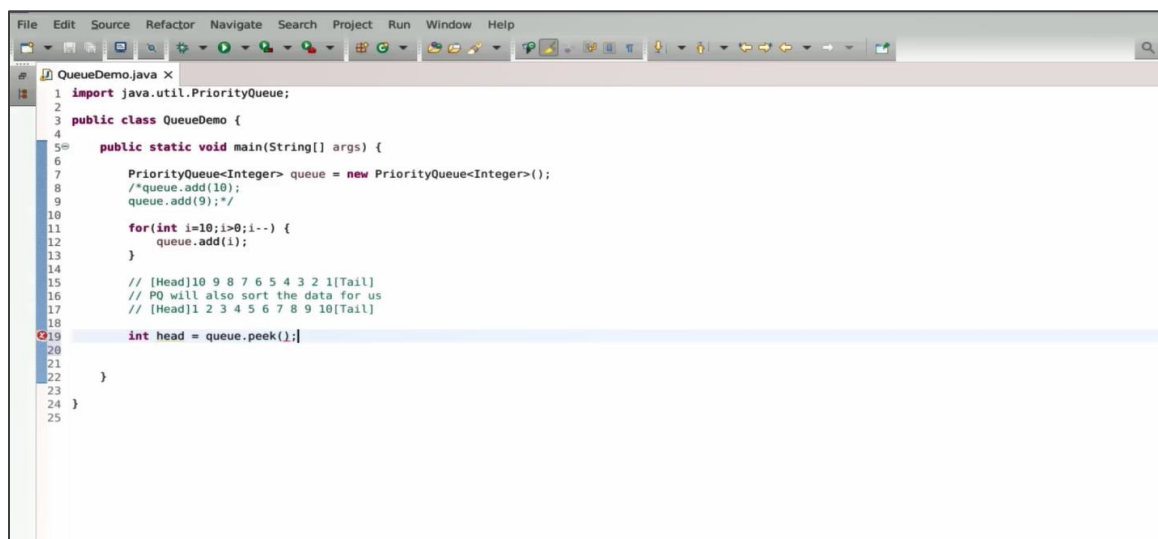
1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15     }
16 }
17
18 }
19

```

This creates a queue with elements from 10 to 1, where 10 is the head and 1 is the tail. The priority queue automatically sorts the data, so the queue will be arranged in ascending order. This means that the head of the queue will be 1, demonstrating the sorted nature of the priority queue.

Step 3: Implement the use of peek and poll

3.1 Write: **int head = queue.peek();**

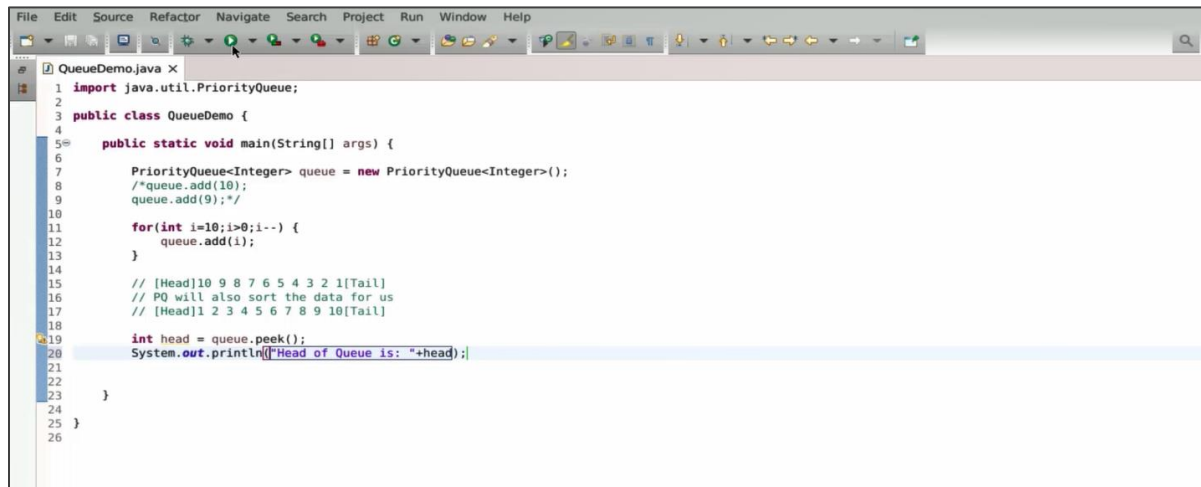


```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         int head = queue.peek();
20
21     }
22 }
23
24 }
25

```

3.2 To retrieve the head of the queue, use the peek method. Write: **“Head of Queue is: “+ head**

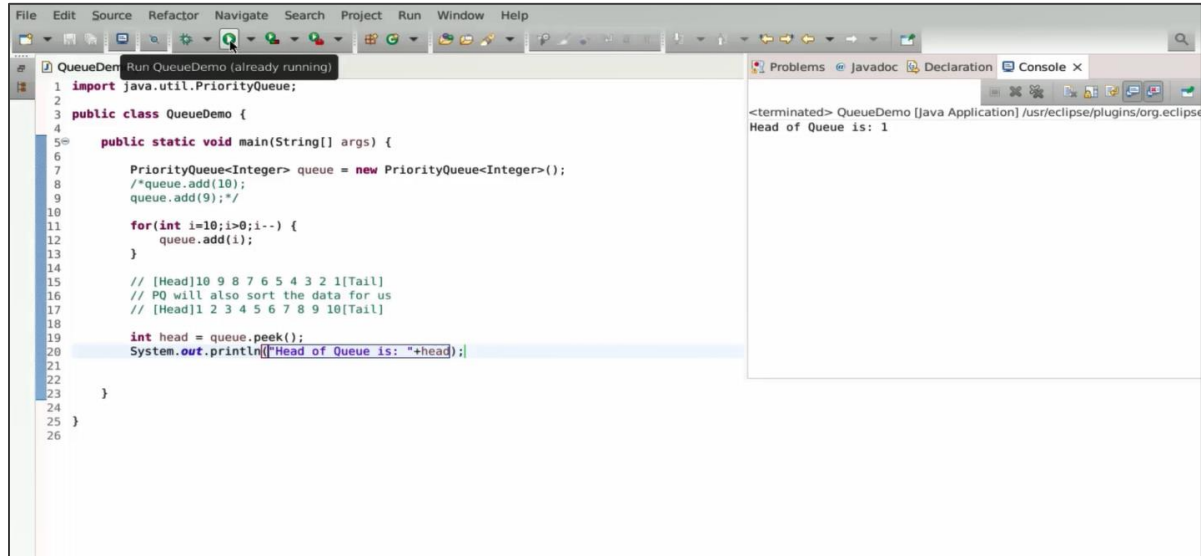


```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22     }
23 }
24
25
26

```

3.3 Output is shown below:
Head of the Queue is: 1



```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22     }
23 }
24
25
26

```

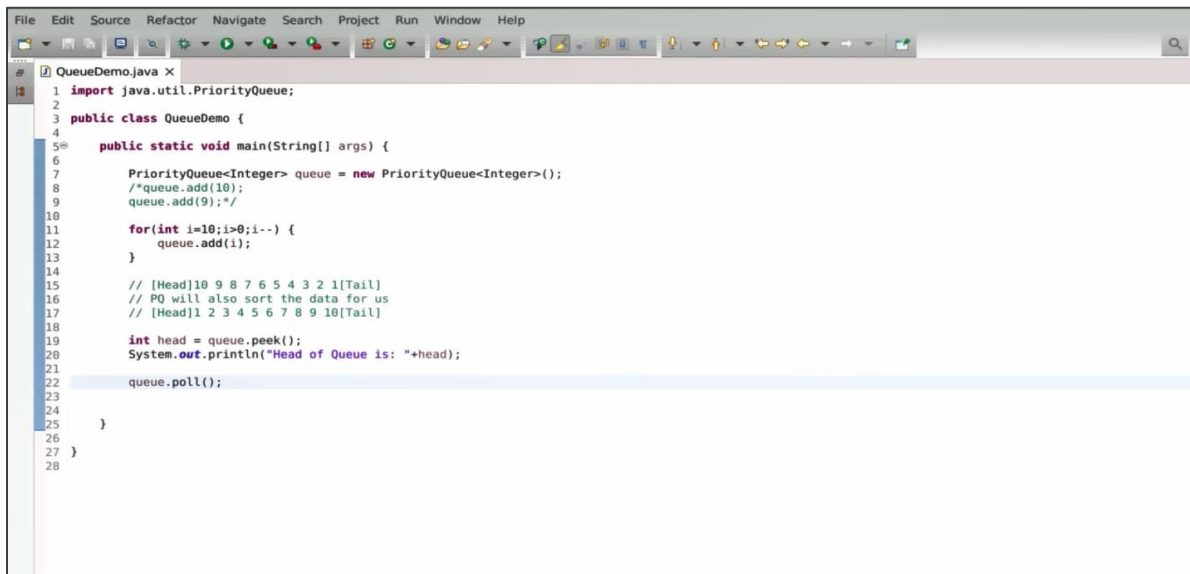
Console Output:

```

<terminated> QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse
Head of Queue is: 1

```

3.4 To remove the head of the queue, write: **queue.poll();**

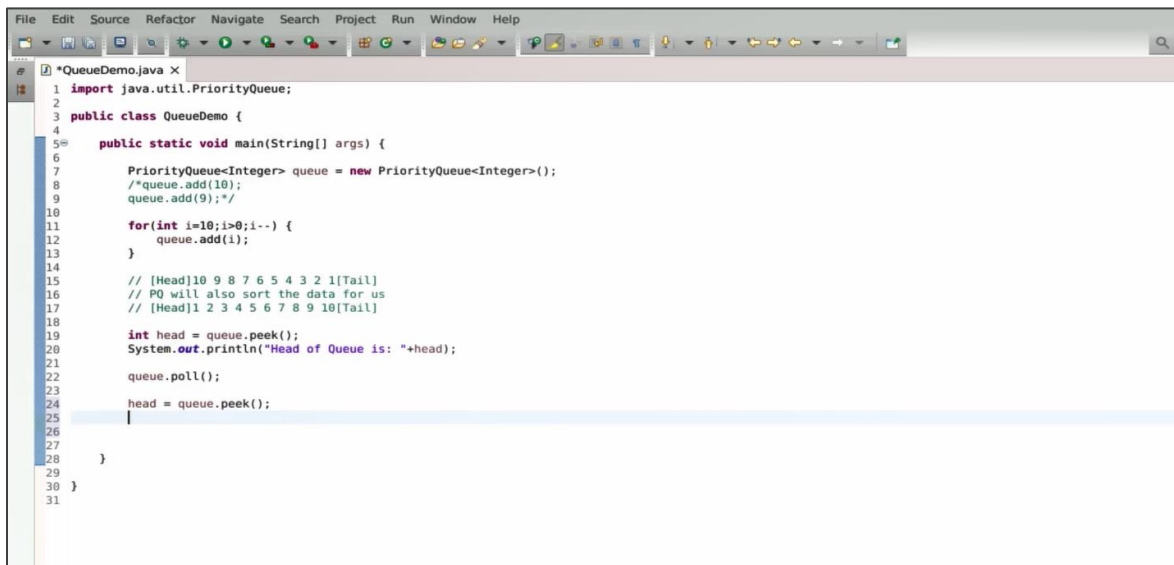


```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24     }
25 }
26
27
28

```

3.5 Retrieve the new head by writing: **head = queue.peek();**



```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25
26     }
27 }
28
29
30
31

```

3.6 Display the new head by writing: "Head of Queue now is: " + head

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);
26
27     }
28 }
29
30 }
31

```

3.7 Output is shown below: Head of Queue now is: 2

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);
26
27     }
28 }
29
30 }
31

```

Console Output:

```

<terminated> QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse
Head of Queue is: 1
Head of Queue now is: 2

```


3.8 Comment out the peek and poll code

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27     }
28 }
29
30 }
31
32 }
33

```

3.9 Write a loop starting with i = 1, i <= 10, and i++ to iterate over the queue

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27         for(int i=1;i<=10;i++) {
28             //
29         }
30
31     }
32 }
33
34 }
35

```

3.10 Print the head of the queue using `queue.peek()`

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27         for(int i=1;i<=10;i++) {
28             System.out.println(queue.peek());
29         }
30     }
31 }
32
33
34
35

```

3.11 Output will display the head as 1 ten times

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27         for(int i=1;i<=10;i++) {
28             System.out.println(queue.peek());
29         }
30     }
31 }
32
33
34
35

```

```

<terminated> QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse
1
1
1
1
1
1
1
1
1
1

```

3.12 Instead of using **peek()**, you can use **queue.poll()** inside the loop to remove the head and display all the elements in the queue

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
7         /*queue.add(10);
8         queue.add(9);*/
9
10        for(int i=10;i>0;i--) {
11            queue.add(i);
12        }
13
14        // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
15        // PQ will also sort the data for us
16        // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
17
18        /*int head = queue.peek();
19        System.out.println("Head of Queue is: "+head);
20
21        queue.poll();
22
23        head = queue.peek();
24        System.out.println("Head of Queue now is: "+head);*/
25
26        for(int i=1;i<=10;i++) {
27            System.out.println(queue.peek());
28            queue.poll();
29        }
30    }
31
32 }
33
34
35
36

```

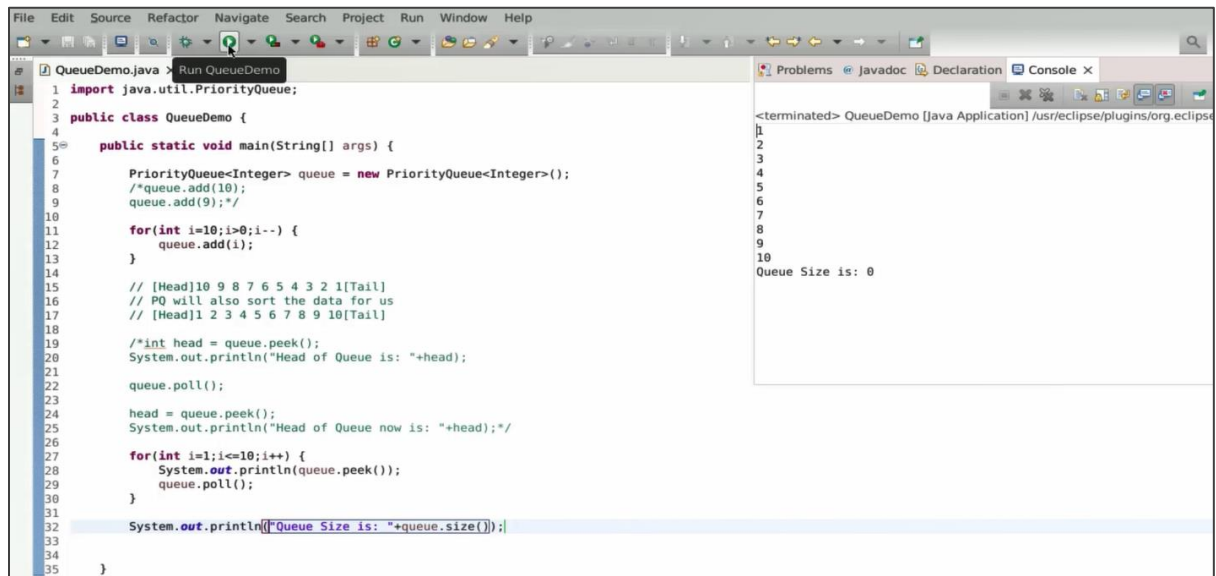
3.13 After the loop, print the size of the queue using "Queue Size is: " + queue.size()

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
7         /*queue.add(10);
8         queue.add(9);*/
9
10        for(int i=10;i>0;i--) {
11            queue.add(i);
12        }
13
14        // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
15        // PQ will also sort the data for us
16        // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
17
18        /*int head = queue.peek();
19        System.out.println("Head of Queue is: "+head);
20
21        queue.poll();
22
23        head = queue.peek();
24        System.out.println("Head of Queue now is: "+head);*/
25
26        for(int i=1;i<=10;i++) {
27            System.out.println(queue.peek());
28            queue.poll();
29        }
30
31        System.out.println("Queue Size is: "+queue.size());
32    }
33
34 }
35
36

```

3.14 Running this code will display that the Queue Size is: 0



The screenshot shows the Eclipse IDE with the file `QueueDemo.java` open. The code defines a `QueueDemo` class with a `main` method. It creates a `PriorityQueue<Integer>` and adds elements 10 and 9. A loop adds elements 1 through 10. Comments indicate the queue's state: `[Head]10 9 8 7 6 5 4 3 2 1[Tail]` and `[Head]1 2 3 4 5 6 7 8 9 10[Tail]`. It prints the head of the queue, polls it, and prints the head again. Finally, it prints the queue size. The console on the right shows the output: `Queue Size is: 0`.

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27         for(int i=1;i<=10;i++) {
28             System.out.println(queue.peek());
29             queue.poll();
30         }
31
32         System.out.println("Queue Size is: "+queue.size());
33     }
34 }

```

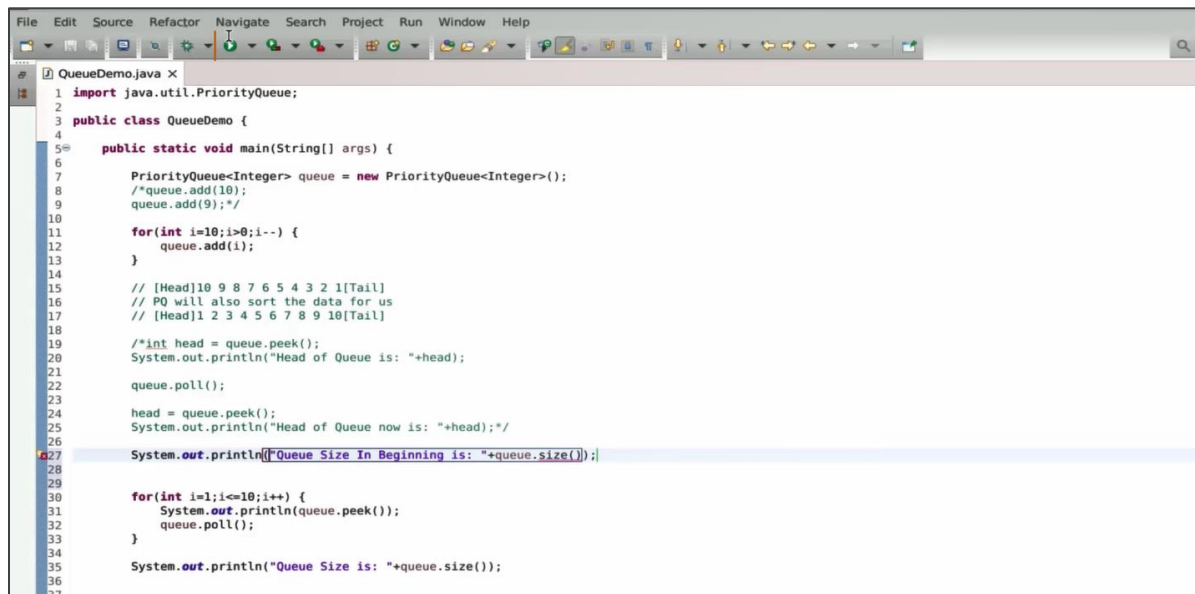
Console Output:

```

<terminated> QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse
1
2
3
4
5
6
7
8
9
10
Queue Size is: 0

```

3.15 Before the loop, print the size of the queue using "Queue Size In the Beginning is: " + queue.size()



The screenshot shows the Eclipse IDE with the modified `QueueDemo.java` file. A new line of code has been added at line 27: `System.out.println("Queue Size In Beginning is: "+queue.size());`. The rest of the code remains the same as in the previous screenshot.

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27         System.out.println("Queue Size In Beginning is: "+queue.size());
28
29         for(int i=1;i<=10;i++) {
30             System.out.println(queue.peek());
31             queue.poll();
32         }
33
34         System.out.println("Queue Size is: "+queue.size());
35     }
36 }
37

```

3.16 Running the program will show that the Queue Size In Beginning is: 10

```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java x Run QueueDemo
1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27         System.out.println("Queue Size In Beginning is: "+queue.size());
28
29         for(int i=1;i<=10;i++) {
30             System.out.println(queue.peek());
31             queue.poll();
32         }
33
34         System.out.println("Queue Size is: "+queue.size());
35
36     }
37 }

```

Console Output:

```

<terminated> QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse
Queue Size In Beginning is: 10
1
2
3
4
5
6
7
8
9
10
Queue Size is: 0

```

3.17 Modify the loop condition from `i <= 10` to `i <= queue.size()`

```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java x
1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27         System.out.println("Queue Size In Beginning is: "+queue.size());
28
29         for(int i=1;i<=queue.size();i++) {
30             System.out.println(queue.peek());
31             queue.poll();
32         }
33
34         System.out.println("Queue Size is: "+queue.size());
35
36     }
37 }

```

3.18 Running this code may give incorrect output. Whenever **poll** is called on the queue inside the loop, the size decreases, which affects the loop condition. Therefore, it is challenging to use **queue.size()** inside the loop for iterative purposes.

The screenshot shows an IDE with a file named `QueueDemo.java`. The code is as follows:

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27         System.out.println("Queue Size In Beginning is: "+queue.size());
28
29         for(int i=1;i<=queue.size();i++) {
30             System.out.println(queue.peek());
31             queue.poll();
32         }
33
34         System.out.println("Queue Size is: "+queue.size());
35     }
36 }

```

The console output on the right shows:

```

<terminated>- QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse
Queue Size In Beginning is: 10
1
2
3
4
5
Queue Size is: 5

```

Step 4: Implement the use of variable size using queue.size()

4.1 Declare a separate variable size and assign it **queue.size()**

The screenshot shows the same IDE with the modified `QueueDemo.java` file. The code is as follows:

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27         System.out.println("Queue Size In Beginning is: "+queue.size());
28
29         int size = queue.size();
30         for(int i=1;i<=size;i++) {
31             System.out.println(queue.peek());
32             queue.poll();
33         }
34
35         System.out.println("Queue Size is: "+queue.size());
36     }
37 }

```

4.2 Modify the loop condition to `i <= size`

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27         System.out.println("Queue Size In Beginning is: "+queue.size());
28
29         int size = queue.size();
30         for(int i=1;i<=size;i++) {
31             System.out.println(queue.peek());
32             queue.poll();
33         }
34
35         System.out.println("Queue Size is: "+queue.size());
36
37     }
38 }

```

4.3 When processing the queue in an algorithmic approach, if using `poll()` inside the loop, the queue cannot be cleared until the end

```

1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15         // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
16         // PQ will also sort the data for us
17         // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
18
19         /*int head = queue.peek();
20         System.out.println("Head of Queue is: "+head);
21
22         queue.poll();
23
24         head = queue.peek();
25         System.out.println("Head of Queue now is: "+head);*/
26
27         System.out.println("Queue Size In Beginning is: "+queue.size());
28
29         int size = queue.size();
30         for(int i=1;i<=size;i++) {
31             System.out.println(queue.peek());
32             queue.poll();
33         }
34
35         System.out.println("Queue Size is: "+queue.size());
36
37     }
38 }

```

Console Output:

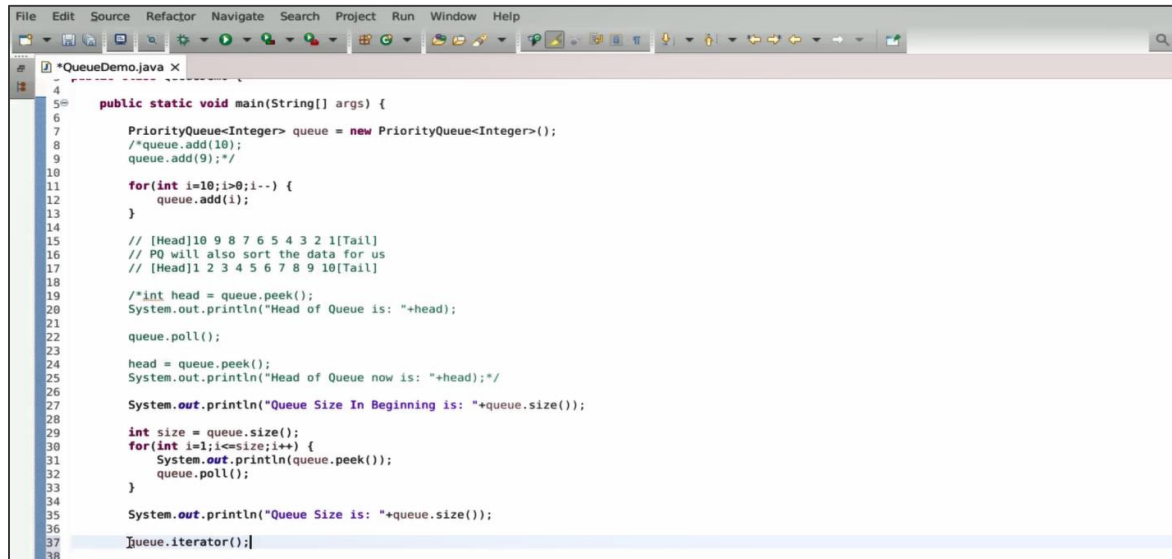
```

<terminated> QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse
Queue Size In Beginning is: 10
1
2
3
4
5
6
7
8
9
10
Queue Size is: 0

```


Step 5: Use the queue.iterator() function with example data

5.1 Write: queue.iterator();



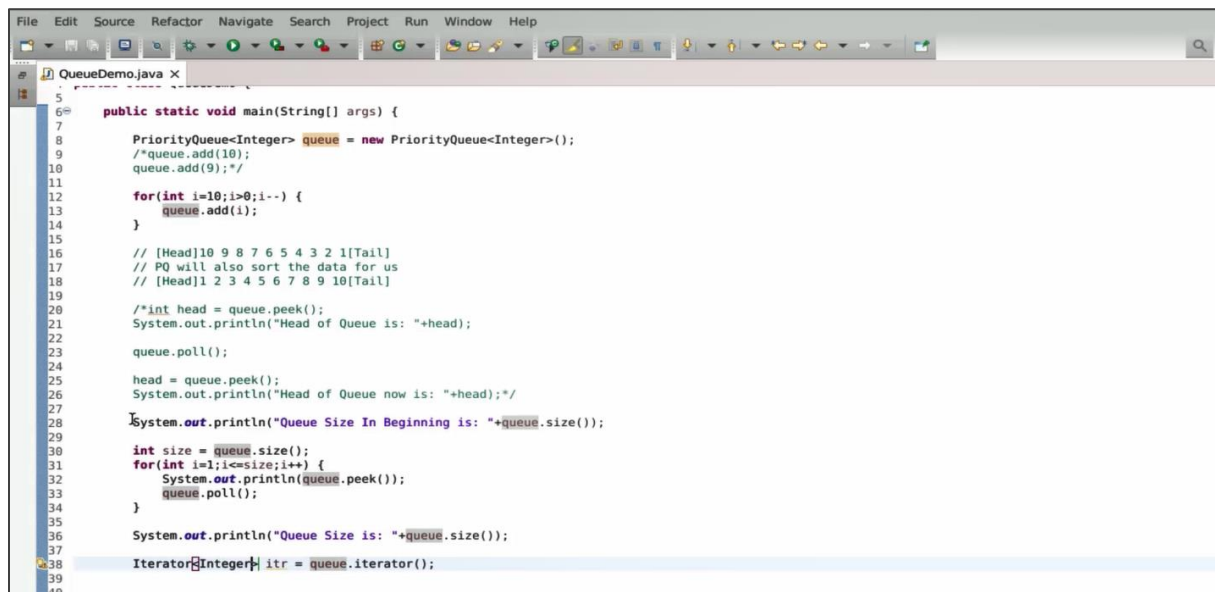
```

1  *QueueDemo.java x
2
3  public static void main(String[] args) {
4
5      PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
6      /*queue.add(10);
7      queue.add(9);*/
8
9      for(int i=10;i>0;i--) {
10         queue.add(i);
11     }
12
13     // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
14     // PQ will also sort the data for us
15     // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
16
17     /*int head = queue.peek();
18     System.out.println("Head of Queue is: "+head);
19
20     queue.poll();
21
22     head = queue.peek();
23     System.out.println("Head of Queue now is: "+head);*/
24
25     System.out.println("Queue Size In Beginning is: "+queue.size());
26
27     int size = queue.size();
28     for(int i=1;i<=size;i++) {
29         System.out.println(queue.peek());
30         queue.poll();
31     }
32
33     System.out.println("Queue Size is: "+queue.size());
34
35     queue.iterator();
36
37 }
38

```

Most collection APIs support iterators.

5.2 Declare an iterator with the data type Integer by writing: **Iterator<Integer> itr = queue.iterator();**



```

1  *QueueDemo.java x
2
3  public static void main(String[] args) {
4
5      PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
6      /*queue.add(10);
7      queue.add(9);*/
8
9      for(int i=10;i>0;i--) {
10         queue.add(i);
11     }
12
13     // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
14     // PQ will also sort the data for us
15     // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
16
17     /*int head = queue.peek();
18     System.out.println("Head of Queue is: "+head);
19
20     queue.poll();
21
22     head = queue.peek();
23     System.out.println("Head of Queue now is: "+head);*/
24
25     System.out.println("Queue Size In Beginning is: "+queue.size());
26
27     int size = queue.size();
28     for(int i=1;i<=size;i++) {
29         System.out.println(queue.peek());
30         queue.poll();
31     }
32
33     System.out.println("Queue Size is: "+queue.size());
34
35     Iterator<Integer> itr = queue.iterator();
36
37 }
38

```


5.3 Comment out the previous code

```

1  *QueueDemo.java X
2
3  public static void main(String[] args) {
4
5      PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
6      /*queue.add(10);
7      queue.add(9);*/
8
9      for(int i=10;i>0;i--) {
10         queue.add(i);
11     }
12
13     // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
14     // PQ will also sort the data for us
15     // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
16
17     /*int head = queue.peek();
18     System.out.println("Head of Queue is: "+head);
19
20     queue.poll();
21
22     head = queue.peek();
23     System.out.println("Head of Queue now is: "+head);*/
24
25     /*System.out.println("Queue Size In Beginning is: "+queue.size());
26
27     int size = queue.size();
28     for(int i=1;i<=size;i++) {
29         System.out.println(queue.peek());
30         queue.poll();
31     }
32
33     System.out.println("Queue Size is: "+queue.size());*/
34
35     Iterator<Integer> itr = queue.iterator();
36
37 }

```

5.4 Iterate over the queue using a while loop with `itr.hasNext()` as the condition

```

1  *QueueDemo.java X
2
3  PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
4  /*queue.add(10);
5  queue.add(9);*/
6
7  for(int i=10;i>0;i--) {
8      queue.add(i);
9  }
10
11 // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
12 // PQ will also sort the data for us
13 // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
14
15 /*int head = queue.peek();
16 System.out.println("Head of Queue is: "+head);
17
18 queue.poll();
19
20 head = queue.peek();
21 System.out.println("Head of Queue now is: "+head);*/
22
23 /*System.out.println("Queue Size In Beginning is: "+queue.size());
24
25 int size = queue.size();
26 for(int i=1;i<=size;i++) {
27     System.out.println(queue.peek());
28     queue.poll();
29 }
30
31 System.out.println("Queue Size is: "+queue.size());*/
32
33 Iterator<Integer> itr = queue.iterator();
34 while(itr.hasNext()) {
35     |
36 }
37
38 }

```

5.5 Print each element using `itr.next()`

```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java x
1
2
3
4
5
6
7
8 PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
9 /*queue.add(10);
10 queue.add(9);*/
11
12 for(int i=10;i>0;i--) {
13     queue.add(i);
14 }
15
16 // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
17 // PQ will also sort the data for us
18 // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
19
20 /*int head = queue.peek();
21 System.out.println("Head of Queue is: "+head);
22
23 queue.poll();
24
25 head = queue.peek();
26 System.out.println("Head of Queue now is: "+head);*/
27
28 /*System.out.println("Queue Size In Beginning is: "+queue.size());
29
30 int size = queue.size();
31 for(int i=1;i<=size;i++) {
32     System.out.println(queue.peek());
33     queue.poll();
34 }
35
36 System.out.println("Queue Size is: "+queue.size());*/
37
38
39 Iterator<Integer> itr = queue.iterator();
40 while(itr.hasNext()) {
41     System.out.println(itr.next());
42 }
43

```

The iteration will display the data in an unordered manner.

```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java x Run QueueDemo
1
2
3
4
5
6
7
8
9
10
11
12 for(int i=10;i>0;i--) {
13     queue.add(i);
14 }
15
16 // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
17 // PQ will also sort the data for us
18 // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
19
20 /*int head = queue.peek();
21 System.out.println("Head of Queue is: "+head);
22
23 queue.poll();
24
25 head = queue.peek();
26 System.out.println("Head of Queue now is: "+head);*/
27
28 /*System.out.println("Queue Size In Beginning is: "+queue.size());
29
30 int size = queue.size();
31 for(int i=1;i<=size;i++) {
32     System.out.println(queue.peek());
33     queue.poll();
34 }
35
36 System.out.println("Queue Size is: "+queue.size());*/
37
38
39 Iterator<Integer> itr = queue.iterator();
40 while(itr.hasNext()) {
41     System.out.println(itr.next());
42 }
43

```

terminated-> QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse

```

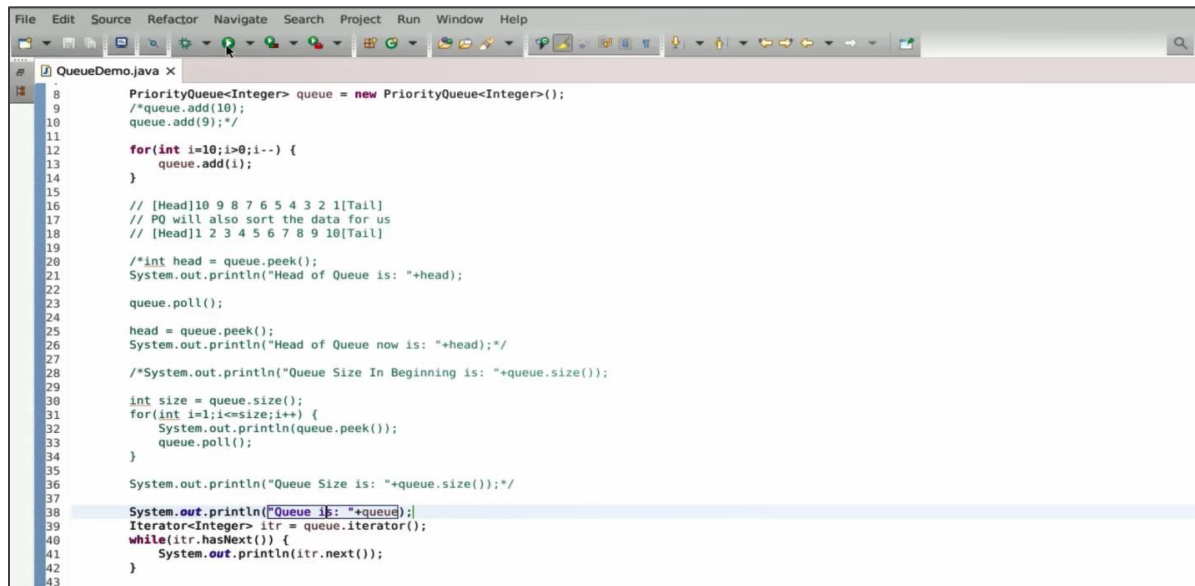
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

```

The priority queue automatically sorts the data, allowing you to process the queue using a first-in-first-out approach with sorted data.

Step 6: Implement of queue prioritization

6.1 When trying to print the queue, for example, write: "Queue is: " + queue

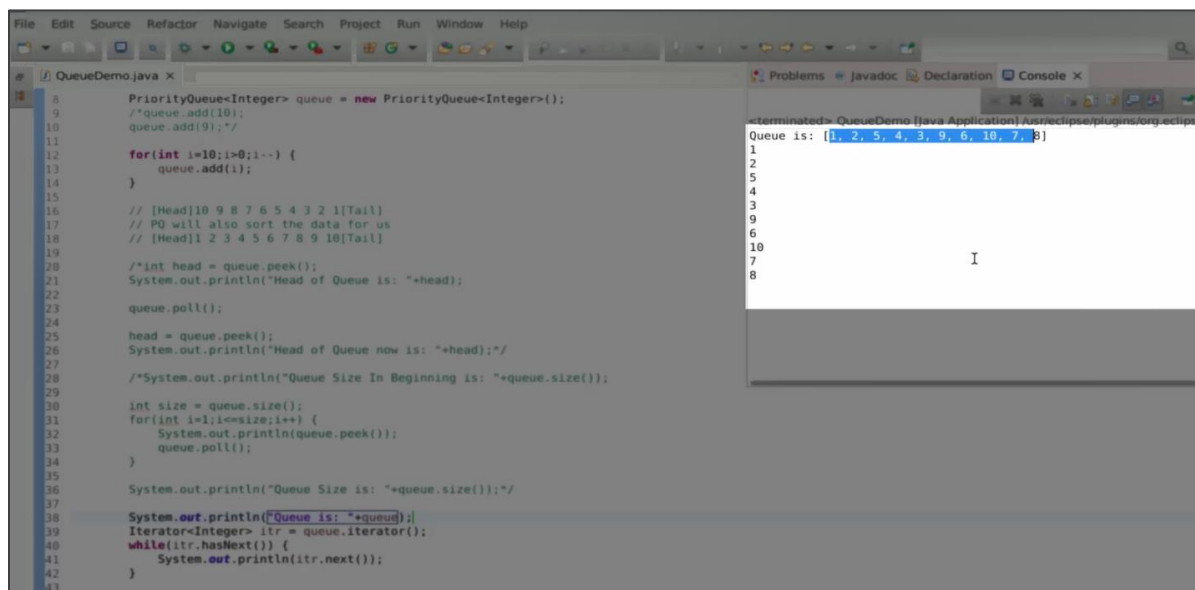


```

8      PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
9      /*queue.add(10);
10     queue.add(9);*/
11
12     for(int i=10;i>0;i--) {
13         queue.add(i);
14     }
15
16     // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
17     // PQ will also sort the data for us
18     // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
19
20     /*int head = queue.peek();
21     System.out.println("Head of Queue is: "+head);
22
23     queue.poll();
24
25     head = queue.peek();
26     System.out.println("Head of Queue now is: "+head);*/
27
28     /*System.out.println("Queue Size In Beginning is: "+queue.size());
29
30     int size = queue.size();
31     for(int i=1;i<=size;i++) {
32         System.out.println(queue.peek());
33         queue.poll();
34     }
35
36     System.out.println("Queue Size is: "+queue.size());*/
37
38     System.out.println("Queue is: " + queue);
39     Iterator<Integer> itr = queue.iterator();
40     while(itr.hasNext()) {
41         System.out.println(itr.next());
42     }
43

```

6.2 The data stored in the queue will be displayed in an unordered arrangement



```

8      PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
9      /*queue.add(10);
10     queue.add(9);*/
11
12     for(int i=10;i>0;i--) {
13         queue.add(i);
14     }
15
16     // [Head]10 9 8 7 6 5 4 3 2 1[Tail]
17     // PQ will also sort the data for us
18     // [Head]1 2 3 4 5 6 7 8 9 10[Tail]
19
20     /*int head = queue.peek();
21     System.out.println("Head of Queue is: "+head);
22
23     queue.poll();
24
25     head = queue.peek();
26     System.out.println("Head of Queue now is: "+head);*/
27
28     /*System.out.println("Queue Size In Beginning is: "+queue.size());
29
30     int size = queue.size();
31     for(int i=1;i<=size;i++) {
32         System.out.println(queue.peek());
33         queue.poll();
34     }
35
36     System.out.println("Queue Size is: "+queue.size());*/
37
38     System.out.println("Queue is: " + queue);
39     Iterator<Integer> itr = queue.iterator();
40     while(itr.hasNext()) {
41         System.out.println(itr.next());
42     }
43

```

Queue is: [1, 2, 5, 4, 3, 9, 6, 10, 7, 8]

By following these steps, you'll gain skills to manage and prioritize data efficiently using a PriorityQueue, which automatically sorts elements for processing.