

## 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

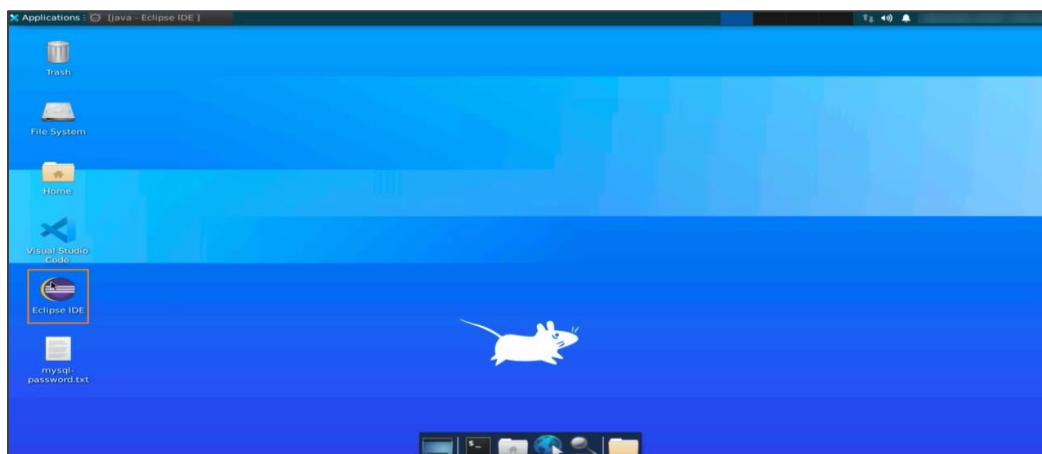
**Prerequisites:** None

Steps to be followed:

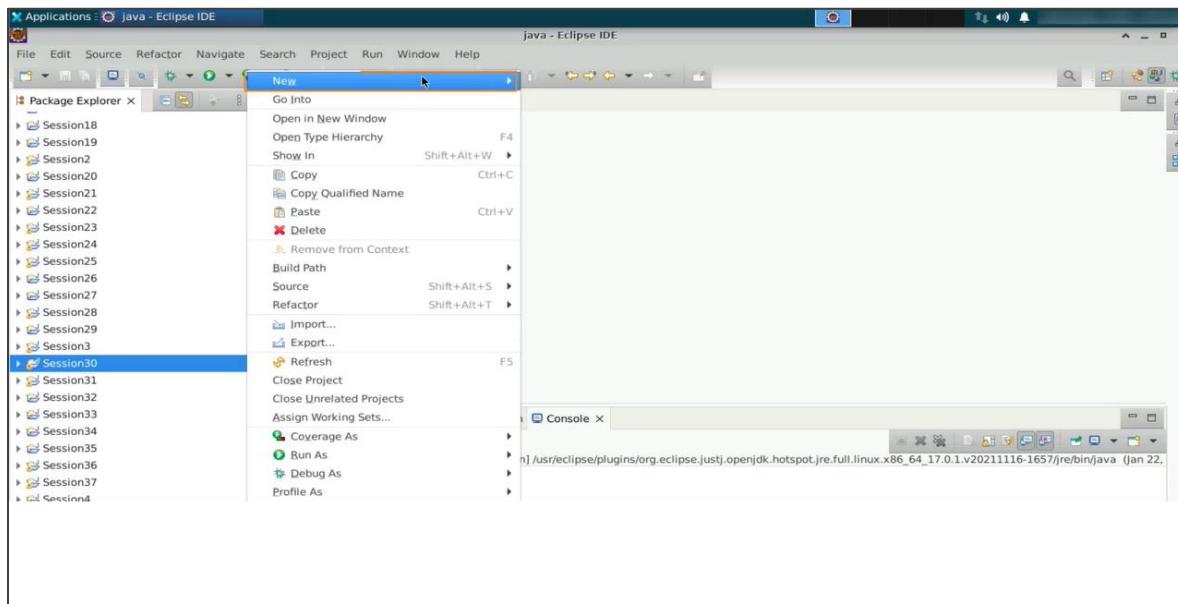
1. Create a new project
2. Add data to the queue using the add method
3. Implement the use of peek and poll methods
4. Implement the use of the variable size using the `queue.size()` method
5. Use the `queue.iterator()` function with example data
6. Implement queue prioritization

#### Step 1: Create a new project

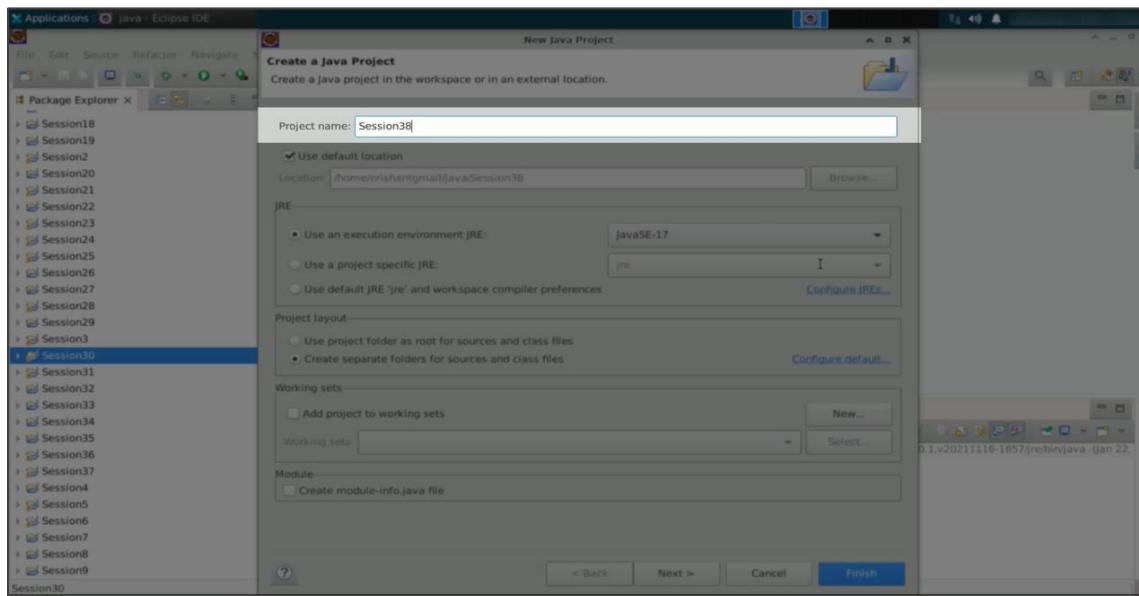
1.1 To create a queue, open the **Eclipse IDE**



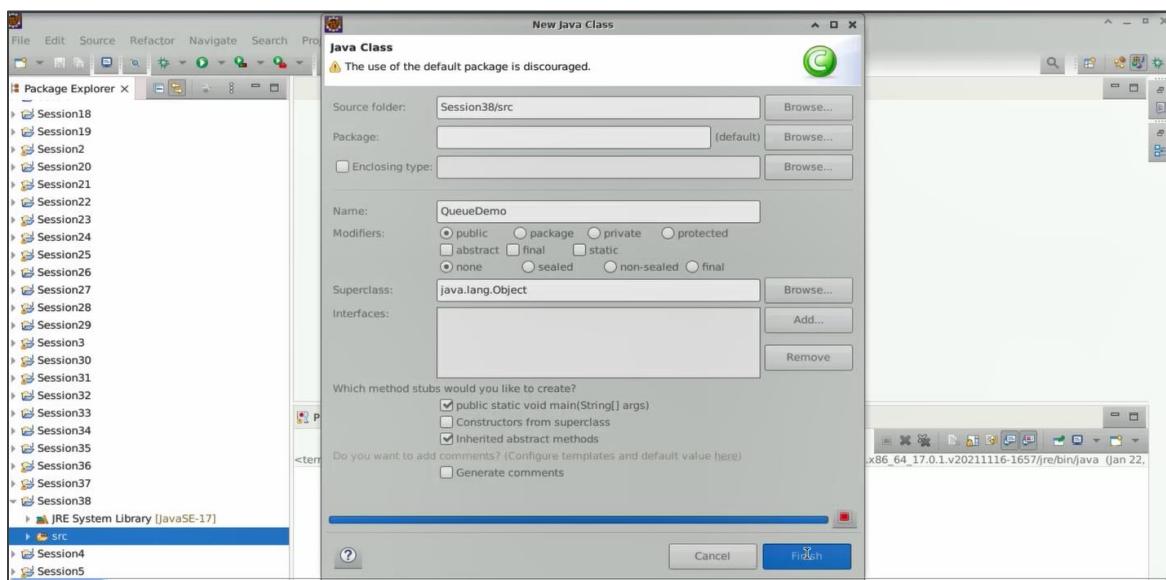
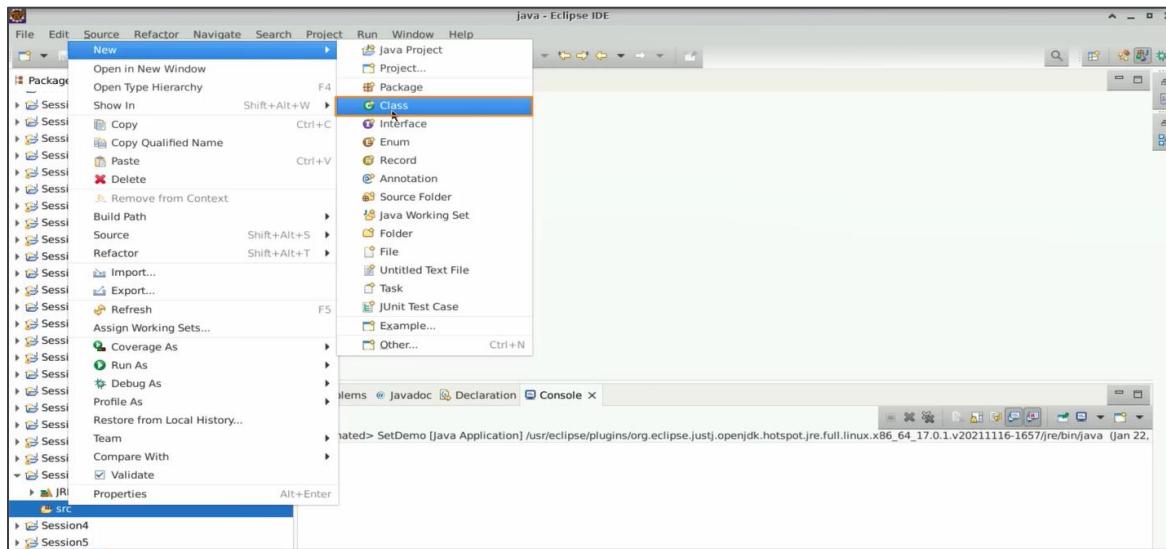
## 1.2 Create a new Java project



## 1.3 Name the new project as **Session38**

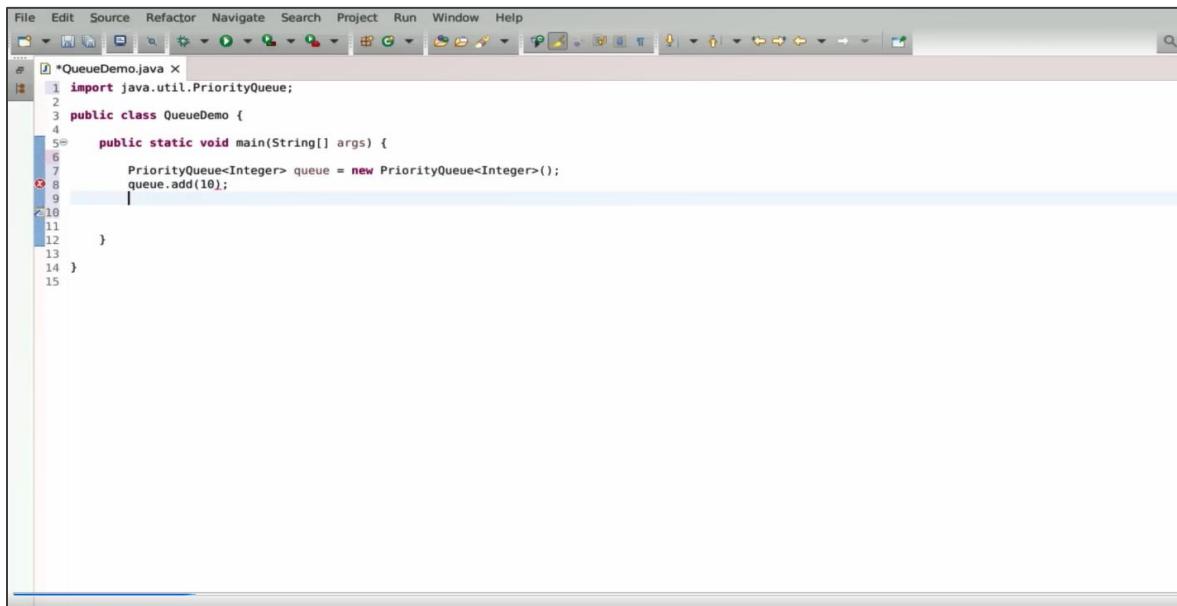


**1.4 Right-click on Session38 and create a new class called QueueDemo with the main method**



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

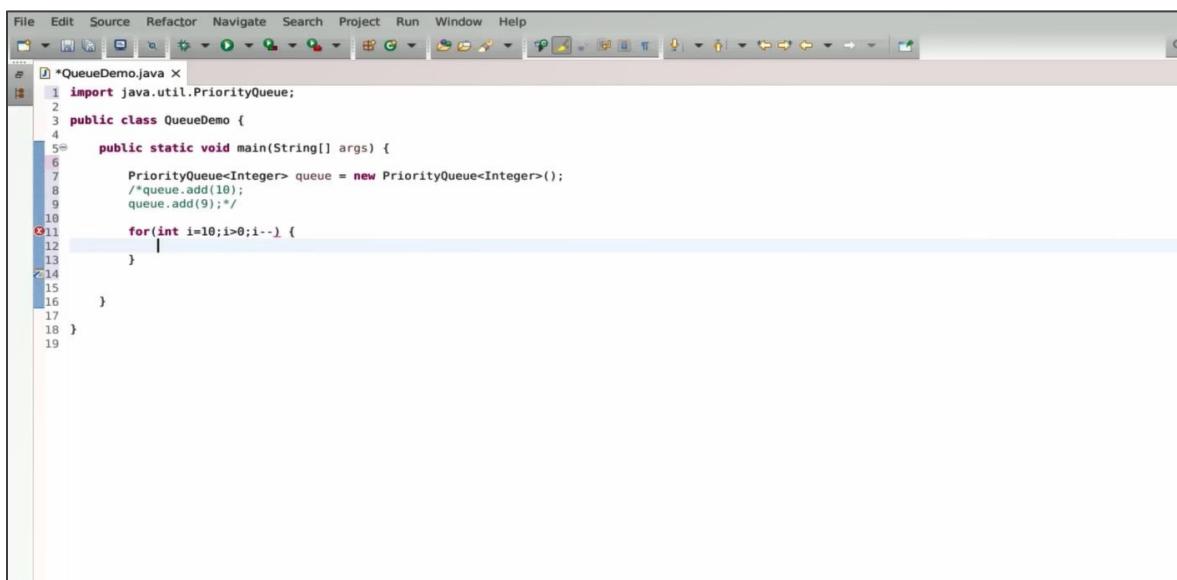
2.1 To add data to the queue, use the add method like lists or sets. Write: `queue.add(10);`



The screenshot shows a Java code editor window with the file "QueueDemo.java" open. The code creates a Priority Queue and adds the integer value 10 to it. A cursor is visible at the end of the line where the queue.add(10) method call is located.

```
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
10    }
11
12 }
13
14 }
15
```

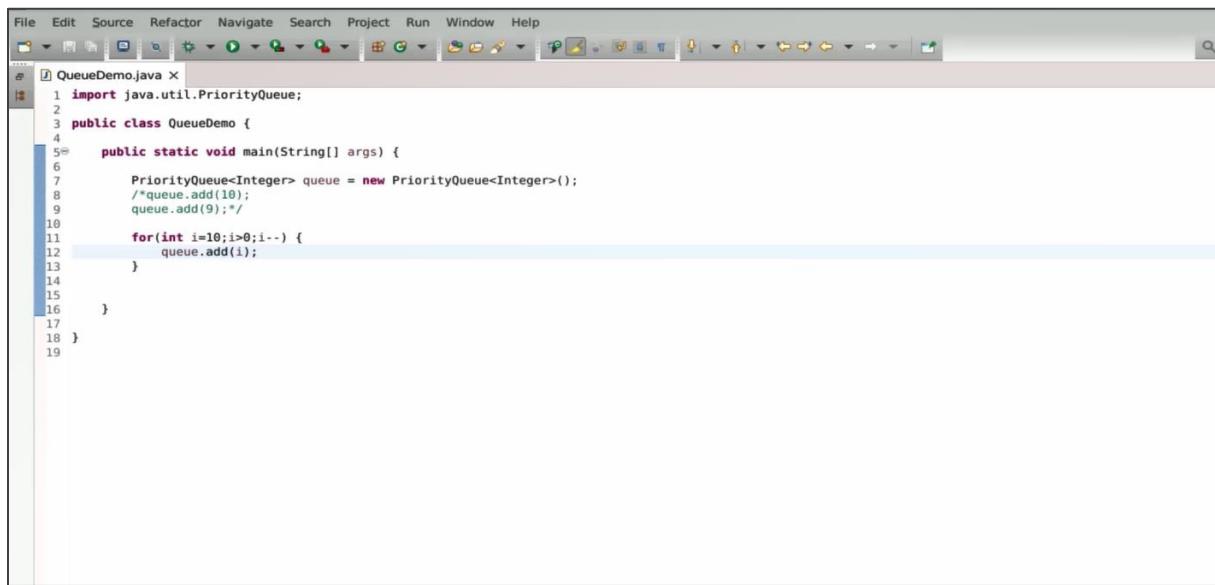
2.2 Add more data to the queue, such as 9, or alternatively use a loop to add 10 more numbers



The screenshot shows the same Java code editor window, but now it includes a for loop that adds integers from 10 down to 1 to the queue. The cursor is positioned at the start of the loop's opening brace.

```
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
13     }
14
15
16 }
17
18 }
```

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



```

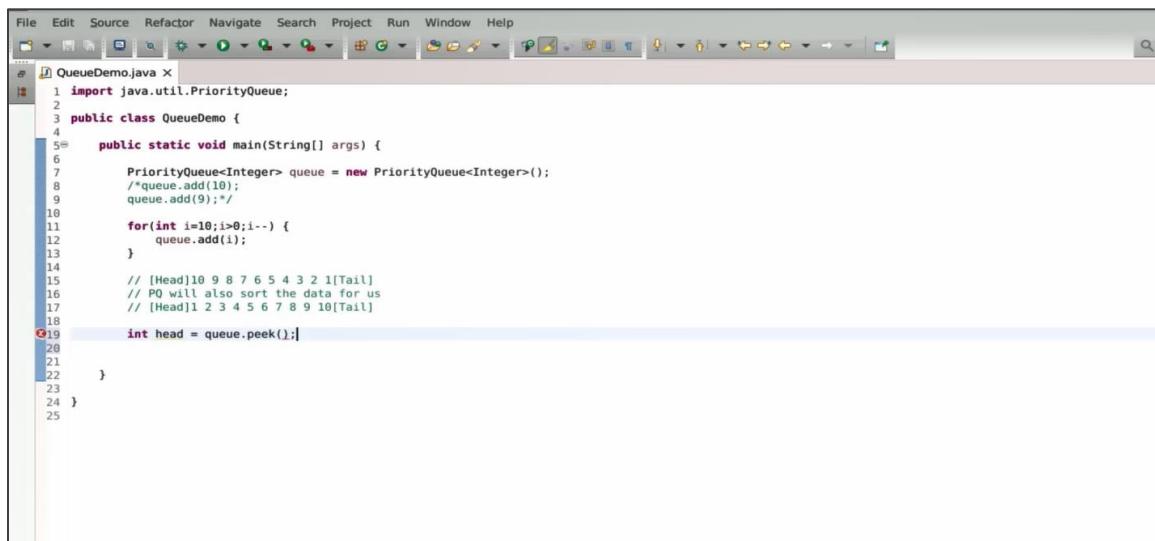
File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java X
1 import java.util.PriorityQueue;
2
3 public class QueueDemo {
4
5     public static void main(String[] args) {
6
7         PriorityQueue<Integer> queue = new PriorityQueue<Integer>();
8         /*queue.add(10);
9         queue.add(9);*/
10
11         for(int i=10;i>0;i--) {
12             queue.add(i);
13         }
14
15     }
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();`



```

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
21
22     }
23
24 }
25

```

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

```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java ×
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**

```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo Run QueueDemo (already running)
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

```

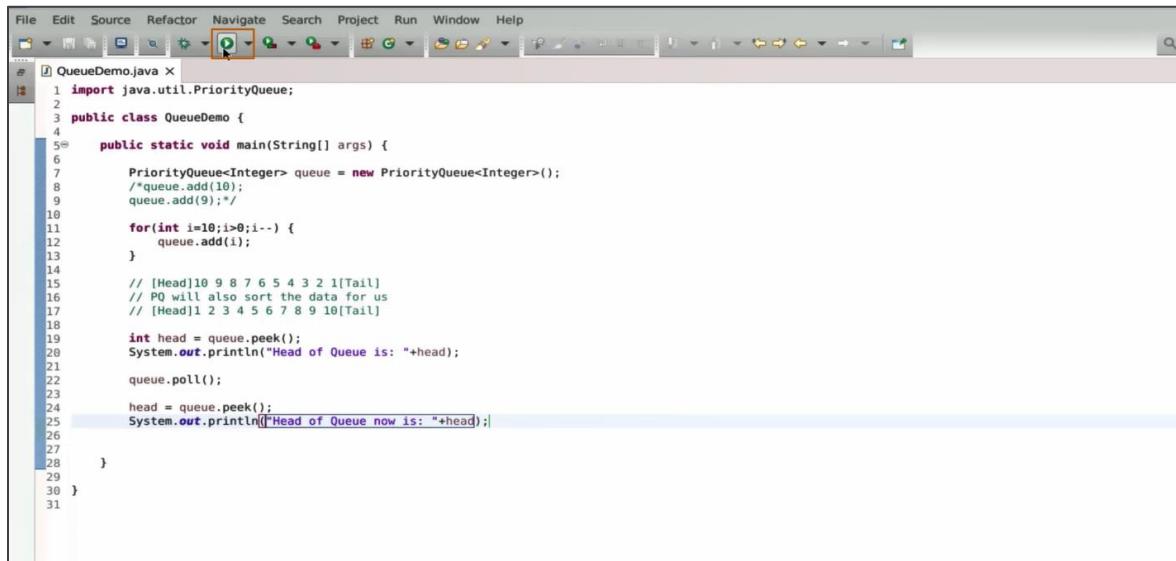
3.4 To remove the head of the queue, write: **queue.poll();**

```
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
25    }
26
27 }
28
```

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

```
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
26
27 }
28
29 }
30
```

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



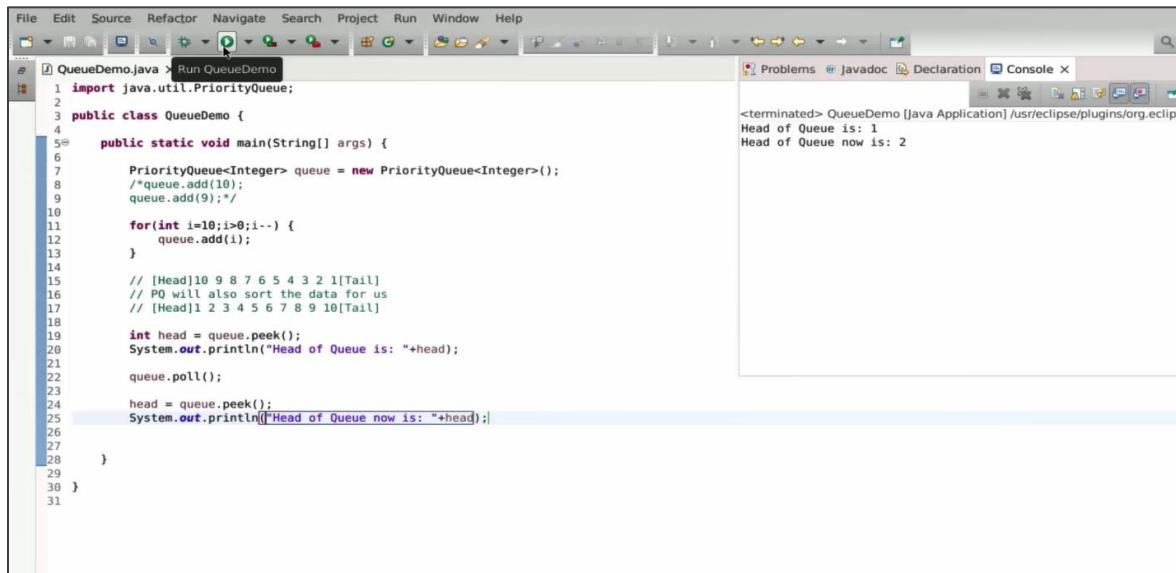
```

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     }
28
29
30 }
31

```

### 3.7 Output is shown below:

**Head of Queue now is: 2**



```

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     }
28
29
30 }
31

```

Console View Output:

```

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

```

### 3.8 Comment out the peek and poll code

```

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
28
29
30     }
31
32 }
33

```

### 3.9 Write a loop starting with $i = 1$ , $i \leq 10$ , and $i++$ to iterate over the queue

```

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         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()`

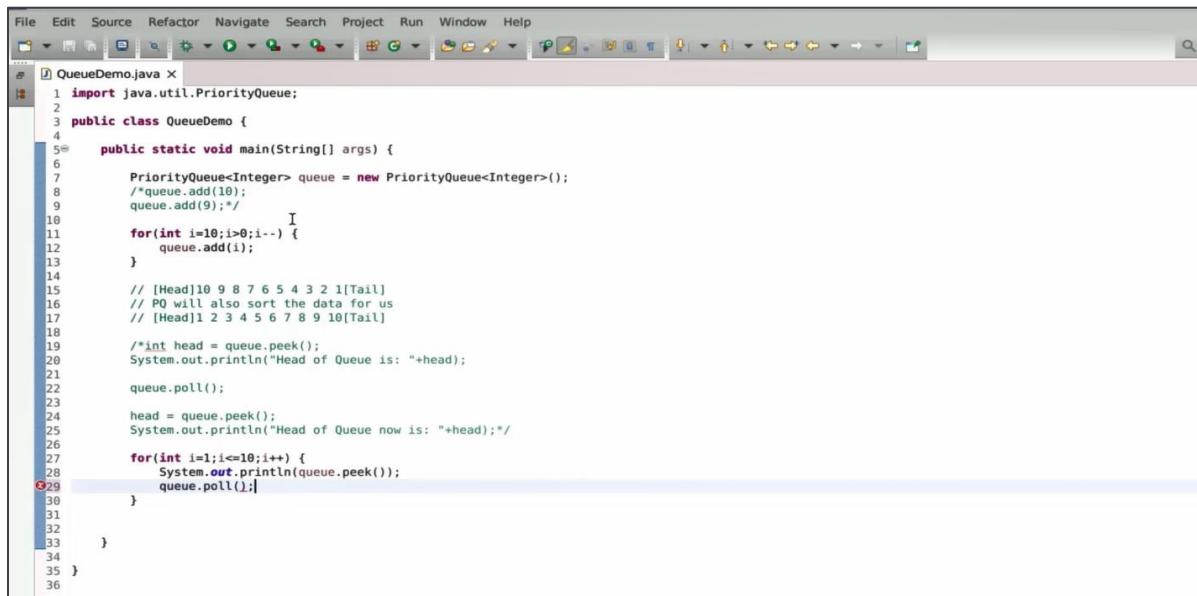
The screenshot shows a Java IDE interface with a toolbar at the top containing various icons for file operations, navigation, and code editing. The left sidebar displays a project tree with a single file named "QueueDemo.java". The main editor area contains the following Java code:

```
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        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

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

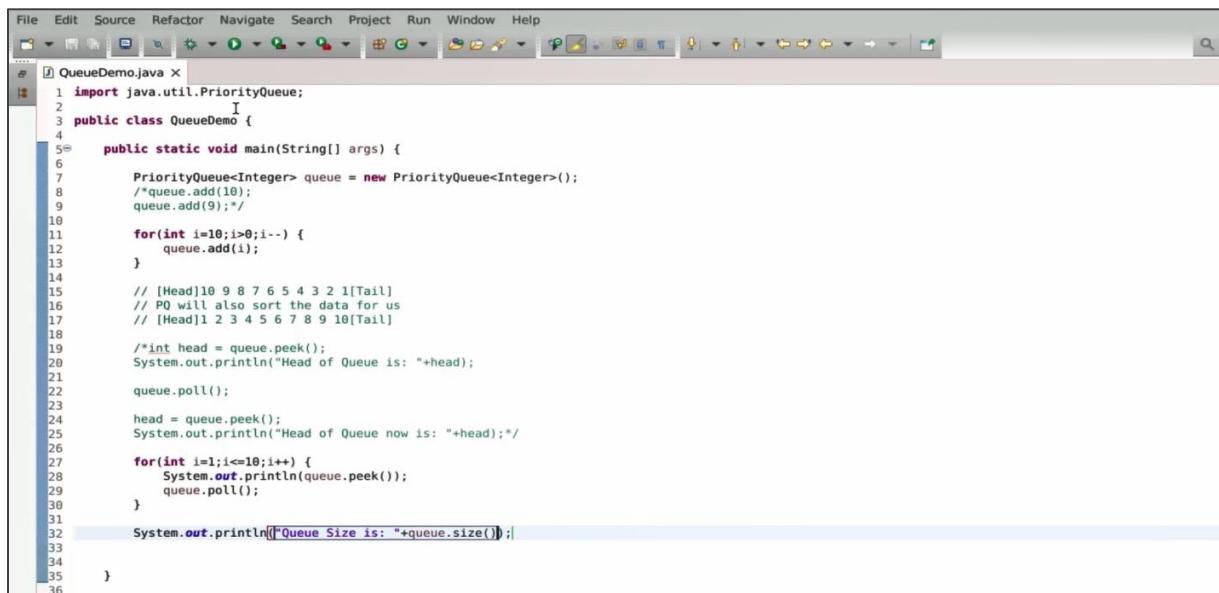


```

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        for(int i=10;i>0;i--) {
11            queue.add(i);
12        }
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    }
33
34
35 }
36

```

3.13 After the loop, print the size of the queue using "Queue Size is: " + 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        for(int i=10;i>0;i--) {
11            queue.add(i);
12        }
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
35 }
36

```

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

The screenshot shows the Eclipse IDE interface. On the left, the code editor displays `QueueDemo.java` with the following content:

```

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    }
35 }
```

On the right, the **Console** view shows the output of the program:

```

<terminated> QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core.prefs
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 interface. On the left, the code editor displays `QueueDemo.java` with the following content:

```

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
30        for(int i=1;i<=10;i++) {
31            System.out.println(queue.peek());
32            queue.poll();
33        }
34
35        System.out.println("Queue Size is: "+queue.size());
36
37    }
38 }
```

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

The screenshot shows the Eclipse IDE interface with the QueueDemo.java file open in the editor. The code creates a Priority Queue and adds elements from 10 down to 1. It then prints the queue size at the beginning and the elements in descending order. The output window shows the queue size as 10 and the elements from 10 to 1.

```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java > 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
30         for(int i=1;i<=10;i++) {
31             System.out.println(queue.peek());
32             queue.poll();
33         }
34
35         System.out.println("Queue Size is: "+queue.size());
36
37

```

<terminated> QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core.prefs  
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 $i \leq 10$ to $i \leq \text{queue.size()}$

The screenshot shows the Eclipse IDE interface with the QueueDemo.java file open in the editor. The code has been modified to use the queue.size() method as the loop condition in the second for loop. The output window shows the queue size as 10 and the elements from 10 to 1.

```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java > 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
30         for(int i=1;i<=queue.size();i++) {
31             System.out.println(queue.peek());
32             queue.poll();
33         }
34
35         System.out.println("Queue Size is: "+queue.size());
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.

```

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
30         for(int i=1;i<queue.size();i++) {
31             System.out.println(queue.peek());
32             queue.poll();
33         }
34
35         System.out.println("Queue Size is: "+queue.size());
36
37

```

terminated> QueueDemo [Java Application] Aus/eclipse/plugins/org.eclipse.jdt.core/compiler.jar  
Queue Size In Beginning is: 10  
I  
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()**

```

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         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

```

## 4.2 Modify the loop condition to $i \leq size$

```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java X 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         // [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

```

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

```

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         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

```

Output in Console:

```

<terminated> QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/compiler.jar
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();`

```

File Edit Source Refactor Navigate Search Project Run Window Help
*QueueDemo.java X
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     queue.iterator();
38

```

Most collection APIs support iterators.

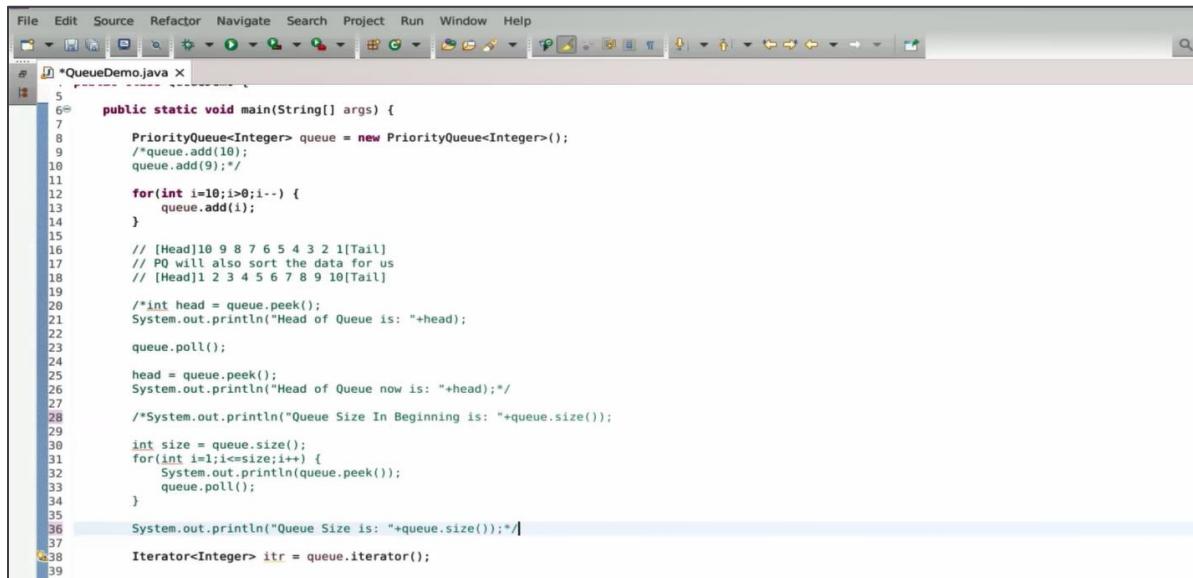
5.2 Declare an iterator with the data type Integer by writing: `Iterator<Integer> itr = queue.iterator();`

```

File Edit Source Refactor Navigate Search Project Run Window Help
* QueueDemo.java X
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     Iterator<Integer> itr = queue.iterator();
38
39
40

```

### 5.3 Comment out the previous code

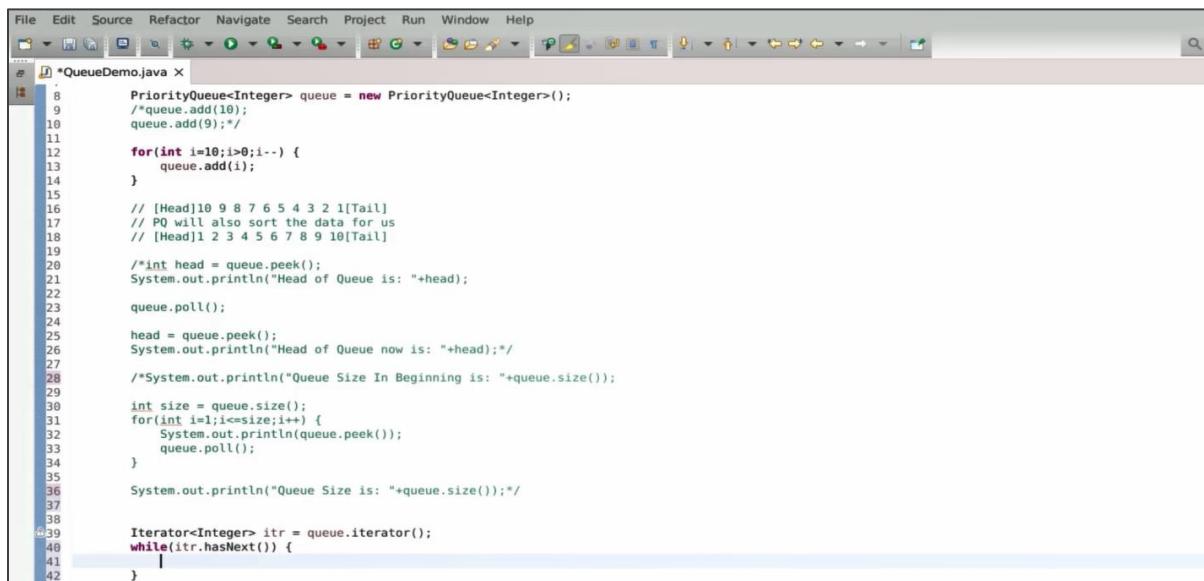


```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java X
5
6  public static void main(String[] args) {
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     Iterator<Integer> itr = queue.iterator();
39
40
41
42

```

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



```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java X
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

```

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

```

File Edit Source Refactor Navigate Search Project Run Window Help
QueueDemo.java X I
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
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

```

terminated> QueueDemo [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/compiler.log

1  
2  
5  
4  
3  
9  
6  
10  
7  
8

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

```

File Edit Source Refactor Navigate Search Project Run Window Help
# QueueDemo.java X
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

```

File Edit Source Refactor Navigate Search Project Run Window Help
# QueueDemo.java X
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

```

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