







<https://raw.githubusercontent.com/vsaravanan/java22/master/src/main/java/com/saravanjs/java22/console/collection/ProrityQueueObject.java>

*/\*  
  
 PriorityQueue is a type of queue in Java that orders its elements according to their natural ordering  
 (if they implement Comparable) or by a Comparator provided at queue construction time.  
  
  
 \*/*@Data  
*class* Employee {  
 *private* String name;  
 *private int* salary;  
  
 *public* Employee(String name, *int* salary) {  
 *this*.name = name;  
 *this*.salary = salary;  
 }  
  
 @Override  
 *public* String toString() {  
 *return* "Employee [name=" + name + ", salary=" + salary + "] \n";  
 }  
  
}  
  
  
*class* EmpComparator *implements Comparator*<Employee> {  
 @Override  
 *public int* compare(Employee o1, Employee o2) {  
 *return* o1.getSalary() - o2.getSalary();  
 }  
}  
  
*public class* ProrityQueueObject {  
 *public static void* main(String[] args) {  
 Employee e1 = *new* Employee("Sarav", 300);  
 Employee e2 = *new* Employee("Raj", 200);  
 Employee e3 = *new* Employee("Ravi", 4000);  
 Employee e4 = *new* Employee("Rahul", 50);  
 Employee e5 = *new* Employee("Juhi", 10);  
  
 EmpComparator comparator = *new* EmpComparator();  
  
 PriorityQueue<Employee > pq = *new* PriorityQueue<>(5, comparator);  
 pq.add(e1);  
 pq.add(e2);  
 pq.add(e3);  
 pq.add(e4);  
 pq.add(e5);  
 pq.add(*new* Employee("F", 3500));  
 pq.add(*new* Employee("G", 50000));  
  
  
 System.*out*.println("peek " + pq.peek());  
  
 System.*out*.println(pq);  
  
 System.*out*.println("for loop will not print in sorted order");  
 *for* (Employee e : pq) {  
 System.*out*.println(e);  
 }  
  
 System.*out*.println("use sorted comparator to get sorted order");  
 pq.stream().sorted(comparator).forEach(System.*out*::println);  
  
 *while* (! pq.isEmpty()) {  
 System.*out*.println("polled " + pq.poll());  
 System.*out*.println(pq);  
 System.*out*.println("-----------------------------------------------------");  
 }  
  
 }  
}

peek Employee [name=Juhi, salary=10]

[Employee [name=Juhi, salary=10]

, Employee [name=Rahul, salary=50]

, Employee [name=F, salary=3500]

, Employee [name=Sarav, salary=300]

, Employee [name=Raj, salary=200]

, Employee [name=Ravi, salary=4000]

, Employee [name=G, salary=50000]

]

for loop will not print in sorted order

Employee [name=Juhi, salary=10]

Employee [name=Rahul, salary=50]

Employee [name=F, salary=3500]

Employee [name=Sarav, salary=300]

Employee [name=Raj, salary=200]

Employee [name=Ravi, salary=4000]

Employee [name=G, salary=50000]

use sorted comparator to get sorted order

Employee [name=Juhi, salary=10]

Employee [name=Rahul, salary=50]

Employee [name=Raj, salary=200]

Employee [name=Sarav, salary=300]

Employee [name=F, salary=3500]

Employee [name=Ravi, salary=4000]

Employee [name=G, salary=50000]

polled Employee [name=Juhi, salary=10]

[Employee [name=Rahul, salary=50]

, Employee [name=Raj, salary=200]

, Employee [name=F, salary=3500]

, Employee [name=Sarav, salary=300]

, Employee [name=G, salary=50000]

, Employee [name=Ravi, salary=4000]

]

-----------------------------------------------------

polled Employee [name=Rahul, salary=50]

[Employee [name=Raj, salary=200]

, Employee [name=Sarav, salary=300]

, Employee [name=F, salary=3500]

, Employee [name=Ravi, salary=4000]

, Employee [name=G, salary=50000]

]

-----------------------------------------------------

polled Employee [name=Raj, salary=200]

[Employee [name=Sarav, salary=300]

, Employee [name=Ravi, salary=4000]

, Employee [name=F, salary=3500]

, Employee [name=G, salary=50000]

]

-----------------------------------------------------

polled Employee [name=Sarav, salary=300]

[Employee [name=F, salary=3500]

, Employee [name=Ravi, salary=4000]

, Employee [name=G, salary=50000]

]

-----------------------------------------------------

polled Employee [name=F, salary=3500]

[Employee [name=Ravi, salary=4000]

, Employee [name=G, salary=50000]

]

-----------------------------------------------------

polled Employee [name=Ravi, salary=4000]

[Employee [name=G, salary=50000]

]

-----------------------------------------------------

polled Employee [name=G, salary=50000]

[]

[*https://raw.githubusercontent.com/vsaravanan/java22/master/src/main/java/console/collection/PriorityQueueExample.java*](https://raw.githubusercontent.com/vsaravanan/java22/master/src/main/java/console/collection/PriorityQueueExample.java)

*class* Task *implements Comparable*<Task> {  
 *private* String name;  
 *private int* priority;  
  
 *public* Task(String name, *int* priority) {  
 *this*.name = name;  
 *this*.priority = priority;  
 }  
  
 *public* String getName() {  
 *return* name;  
 }  
  
 *public int* getPriority() {  
 *return* priority;  
 }  
  
 @Override  
 *public int* compareTo(Task other) {  
 *return* Integer.*compare*(*this*.priority, other.priority);  
 }  
  
 @Override  
 *public* String toString() {  
 *return* "Task{" +  
 "name='" + name + '\'' +  
 ", priority=" + priority +  
 '}';  
 }  
}  
  
*public class* PriorityQueueExample {  
 *public static void* main(String[] args) {  
 *// Create a PriorityQueue* PriorityQueue<Task> taskQueue = *new* PriorityQueue<>();  
  
 *// Add tasks to the queue* taskQueue.add(*new* Task("Write report", 2));  
 taskQueue.add(*new* Task("Attend meeting", 1));  
 taskQueue.add(*new* Task("Complete assignment", 3));  
  
 *// Process tasks in priority order  
 while* (!taskQueue.isEmpty()) {  
 Task task = taskQueue.poll();  
 System.*out*.println("Processing task: " + task);  
 }  
 }  
}

Processing task: Task{name='Attend meeting', priority=1}

Processing task: Task{name='Write report', priority=2}

Processing task: Task{name='Complete assignment', priority=3}