Lab 6

L1 slot

Valiveti manikanta bhuvanesh

19BCD7088

1.

Create the generic interface GenericQueuable<T> that contains the following abstract methods:

- an abstract method insertEnd(T e) which adds the element to the Queue at end.

- an abstract method removeBegin() which removes a element from the begining of the queue.

- an abstract method printQueue() which prints the queue elements from the front of the queue to end.

- an abstract method isQueueEmpty() which returns true if the queue is empty otherwise return false.

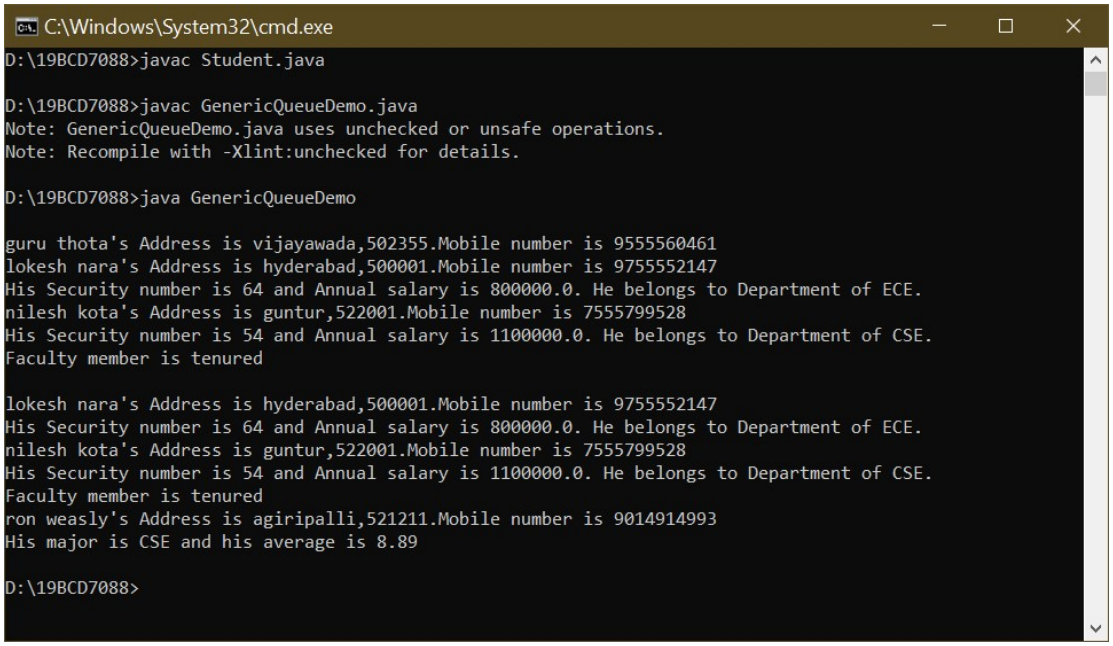
Here 'T' should be bounded by Person and its Children.(Refer Lab3 Exercise Question 5 to know the Person hierarchy).

Create GenericQueue<T> such that it implements GenericQueuable<T>. Write a GenericQueueDemo class to test the operations of GenericQueue class with two different queues.

```
interface GenericQueuable<T>{
	public void insertEnd(T e);
	public void removeBegin();
	public void printQueue();
	public boolean isQueueEmpty();
}
class GenericQueue<T> implements GenericQueuable<T>{
	int front,rear;
	int max;
	T a[];
	GenericQueue(int max){
		this.max=max;
		front=rear=0;
		a=(T[]) new Object[max];
	}
	public void insertEnd(T e){
		if(rear==max){
```

```
                System.out.println("Queue is full");
        }
        else{
                a[rear]=e;
                rear++;
        }
}
public void removeBegin(){
        if(front==rear){
                System.out.println("Queue is empty");
        }
        else{
                for(int j=0;j<rear-1;j++){
                   a[j]=a[j+1];
                }
                rear--;
        }
}
public void printQueue(){
        for(int i=front;i<rear;i++){
                System.out.print(a[i] + " ");
        }
        System.out.println();
}
public boolean isQueueEmpty(){
        if(rear==front){
                return true;
        }
        else{
                return false;
        }
```

```
	}
}
public class GenericQueueDemo{
	public static void main(String[] args) {
		GenericQueue<Person> g = new GenericQueue<Person>(3);
		Person p1=new Person("guru","thota","vijayawada",502355,9555560461L);
		Person p2=new
CollegeEmployee("lokesh","nara","hyderabad",500001,9755552147L,64,800000,"ECE");
		Person p3=new
Faculty("nilesh","kota","guntur",522001,7555799528L,54,1100000,"CSE",true);
		Person p4=new
Student("ron","weasly","agiripalli",521211,9014914993L,"CSE",8.89);
		g.insertEnd(p1);
		g.insertEnd(p2);
		g.insertEnd(p3);
		g.printQueue();
		g.removeBegin();
		g.insertEnd(p4);
		g.printQueue();
	}
}
```



2.a.

Create a generic Map interface MyMap<K,V> that represents a Map structure. K is the type for a key, V is the type of a value. A key is unique in a map and cannot be repeated.

The map contains the following abstract methods.

-an abstract method add(K key, V value) which adds the element to the map

-an abstract method remove(K key) which removes the element with the specified key from the map and returns the value removed.

-a abstract method size() which returns the size of the map

-a abstract method isEmpty() that would return true if the map is empty, and false otherwise.

-a abstract method keys() that returns the list of all keys.

-a abstract method print() that prints all the elements of the map.

2.b.

Create a generic class MyMapImpl that implements the interface MyMap. Use an ArrayList or array to store the keys, and another ArrayList or array to store the values.

2.c.

Create a test class "MyMapTest" that creates two Maps.

Map1: <String, Integer> where the key is a String and the value is an Integer

Map2: <Integer, Double> where the key is a Integer and the value is an Double

Add several elements. Try to add elements with the same key, and check that only one instance is added effectively with no redundancy.

```
interface MyMap<K,V>{
	public void add(K k,V v);
	public void remove(K k);
	public int size();
	public boolean isEmpty();
	public K []keys();
	public void print();
}
class MyMapImpl<K,V> implements MyMap<K,V>{
	K a[];
	V b[];
	int p,max,size,ind;
	boolean o;
	MyMapImpl(int n){
		p=-1;
		max=n;
		size = 0;
		a=(K[]) new Object[n];
		b=(V[]) new Object[n];
	}
	public boolean isPresent(K k){
		for (int i=0;i<size;i++){
			if(k.equals(a[i])){
				o=true;
			}
		}
		if(o){
			return false;
		}
```

```
        else{
                return true;
        }
}
public void add(K k ,V v){
        if(isPresent(k)){
        if(size==max){
                System.out.println("Map is full");
        }
        else{
                p++;
                a[p]=k;
                b[p]=v;
                size ++;
        }
}
else{
        System.out.println(k +" Key is already defined");
}
}
public void remove(K k){
        if(size!=0){
        for(int j=0;j<size;j++){
                if(k.equals(a[j])){
                        ind=j;
                }
        }
        for(int l=ind;l<size-ind;l++){
                a[l]=a[l+1];
                b[l]=b[l+1];
        }
```

```
                size--;
                p--;
        }
        else{
                System.out.println("Map is empty");
        }
        }
        public int size(){
                return size;
        }
        public boolean isEmpty(){
                if(size==0){
                        return true;
                }
                else{
                        return false;
                }
        }
        public K []keys(){
                return a;
        }
        public void print(){
                for(int m=0;m<size;m++){
                        System.out.println(a[m] + " : " + b[m]);
                }
        }
}
public class MyMapTest{
        public static void main(String[] args) {
                MyMapImpl<String, Integer> r = new MyMapImpl<String, Integer>(3);
                MyMapImpl<Integer, Double> u = new MyMapImpl<Integer, Double>(3);
```

```
                r.add("bhuvanesh",22222);
                r.add("guru",52652);
                r.add("manish",85642);
                r.print();
                System.out.println("After removeing one element");
                r.remove("guru");
                r.print();
                r.add("manish",1245);
                System.out.println();
                u.add(7088,665.265);
                u.add(7034,523.245);
                u.add(7110,745.261);
                u.print();
                System.out.println("After removeing one element");
                u.remove(7110);
                u.print();
                System.out.println();
                System.out.println("Is map is empty " + u.isEmpty());
        }
}
```