package Collections.Queue;  
  
import java.util.ArrayDeque;  
import java.util.\*;  
import java.util.Iterator;  
  
public class ArrayDequeClass {  
 public static void main(String args[]) {  
 ArrayDeque<String> animals = new ArrayDeque<>();  
 animals.offer("Giraffe");  
 animals.add("Dog");  
 animals.addFirst("Cat");  
 animals.addLast("Horse");  
 animals.offer("Giraffe");  
 animals.offer("Dog");  
 animals.offerFirst("Cat");  
 animals.offerLast("Rhino");  
 boolean offerValue = animals.offer("Giraffe");  
 boolean addValue = animals.add("Giraffe");  
 System.*out*.println("Animals: " + animals + "offerValue " + offerValue + "addValue" + addValue);  
  
 String firstElement = animals.getFirst();  
 System.*out*.println("Animals after firstElement " + animals + "firstElement " + firstElement);  
  
 String LastElement = animals.getLast();  
 System.*out*.println("Animals after LastElement " + animals + "LastElement " + LastElement);  
  
 String elementPeek = animals.peek();  
 System.*out*.println("Animals after elementPeek " + animals + "elementPeek " + elementPeek);  
  
 String firstElementPeek = animals.peekFirst();  
 System.*out*.println("Animals after firstElementPeek " + animals + "firstElementPeek " + firstElement);  
  
 String lastElementPeek = animals.peekLast();  
 System.*out*.println("Animals after lastElementPeek " + animals + "lastElementPeek " + lastElementPeek);  
  
 String elementRemove = animals.remove();  
 System.*out*.println("Animals after elementRemove " + animals + "elementRemove " + elementRemove);  
  
 String firstElementRemove = animals.removeFirst();  
 System.*out*.println("Animals after firstElementRemove " + animals + "firstElementRemove " + firstElementRemove);  
  
 String lastElementRemove = animals.removeLast();  
 System.*out*.println("Animals after lastElementRemove " + animals + "lastElementRemove " + lastElementRemove);  
  
 String elementpop = animals.pop();  
 System.*out*.println("Animals after elementpop " + animals + "elementpop " + elementpop);  
  
 animals.push("Tiger");  
 System.*out*.println("Animals after push " + animals);  
  
 String elementPoll = animals.poll();  
 System.*out*.println("Animals after Elementpoll " + animals + "Elementpoll " + elementPoll);  
  
 String firstElementpoll = animals.pollFirst();  
 System.*out*.println("Animals after firstElementpoll " + animals + "firstElementpoll " + firstElementpoll);  
  
 String lastElementpoll = animals.pollLast();  
 System.*out*.println("Animals after lastElementpoll " + animals + "lastElementpoll " + lastElementpoll);  
  
 String element = animals.element();  
 System.*out*.println("Animals after element " + animals + "element " + element);  
  
 Boolean firstOcurenceElement = animals.removeFirstOccurrence("Giraffe");  
 System.*out*.println("Animals after firstOcurenceElement " + animals + "firstOcurenceElement " + firstOcurenceElement);  
  
 Boolean lastOcurenceElement = animals.removeFirstOccurrence("Giraffe");  
 System.*out*.println("Animals after lastOcurenceElement " + animals + "lastOcurenceElement " + lastOcurenceElement);  
  
 Iterator<String> animlasDescendingIterator = animals.descendingIterator();  
 while (animlasDescendingIterator.hasNext()) {  
 if (animlasDescendingIterator.next().equals("Rhino")) {  
 animlasDescendingIterator.remove();  
 }  
 }  
 System.*out*.println("Animals after DescendingIterator " + animals);  
  
// Equals test  
  
 ArrayDeque<String> animalClone = animals;  
 animalClone.add("Rhino");  
 System.*out*.println("Animals after animalClone " + animals);  
 Boolean hasAnimalEquals = animals.equals(animalClone);  
 System.*out*.println("hasAnimalEquals " + hasAnimalEquals);  
  
 ArrayDeque<String> animalClone1 = animals;  
 animalClone1.add("Horse");  
 animalClone1.add("Dog");  
 animalClone1.add("Rhino");  
 Boolean hasAnimalEquals1 = animals.equals(animalClone);  
 System.*out*.println("hasAnimalEquals " + hasAnimalEquals1);  
 System.*out*.println("hashCode() " + animals.hashCode());  
  
 //unimplemented Collections methods  
  
 ArrayDeque<String> widerLivingCreatures = new ArrayDeque<>();  
 widerLivingCreatures.add("Man");  
 widerLivingCreatures.add("Snake");  
 widerLivingCreatures.add("fishes");  
 widerLivingCreatures.add("Dinoaueraus");  
  
 Boolean removeif= widerLivingCreatures.remove("Man");  
 System.*out*.println("widerLivingCreatures " + widerLivingCreatures+"removeif "+removeif);  
  
 Boolean hasAnimalContains = animals.contains("Dog");  
 System.*out*.println("hasAnimalContains " + hasAnimalContains);  
  
 animals.addAll(widerLivingCreatures);  
 System.*out*.println("Animals after AddAll " + animals);  
  
 animals.retainAll(widerLivingCreatures);  
 System.*out*.println("Animals after retainAll " + animals);  
  
 animals.removeAll(widerLivingCreatures);  
 System.*out*.println("Animals after removeAll " + animals);  
  
 System.*out*.println("Animals after isEmpty " + animals.isEmpty());  
  
 animals.add("reptiles");  
 animals.add("amphibions");  
 animals.add("birds");  
 animals.add("fishes");  
 animals.add("Other creatures");  
 System.*out*.println("Animals after New Elemets added " + animals);  
 System.*out*.println("Animals after AscendingIterator " + animals);  
  
 String[] finalAnimalsArray = animals.toArray(new String[0]);  
 for (String finalAnimal : finalAnimalsArray) {  
 System.*out*.println(" " + finalAnimal);  
 }  
 Object[] listAarray = animals.toArray();  
 int count = 0;  
 System.*out*.println("ArrayValues");  
 for (Object arrayedValues : listAarray) {  
 System.*out*.println("index " + count + "Arrayvalues " + arrayedValues);  
 count++;  
 }  
  
 System.*out*.println("\n Iterated values");  
 Iterator<String> ascendingIterator=animals.iterator();  
 {  
  
 while (ascendingIterator.hasNext()) {  
 String value = ascendingIterator.next();  
 if (value.equals( "fishes")) {  
 ascendingIterator.remove();  
// ascendingIterator.forEachRemaining(x->System.out.println(x+"Remaining"));  
 }  
 System.*out*.print(" "+value);  
 }  
 }  
  
 // ArrayQueue Speific function  
 System.*out*.println("\n descending values");  
 Iterator<String> descendingIterator = animals.descendingIterator();  
 while (descendingIterator.hasNext()) {  
 String value1= descendingIterator.next();  
 if (value1.equals("amphibions")) {  
 descendingIterator.remove();  
 }  
 System.*out*.print(" "+value1);  
 }  
 System.*out*.println("\n Animals " + animals);  
 animals.clear();  
 System.*out*.println("Animals after clear " + animals);  
  
 }  
}

C:\Users\Roystan\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.3\lib\idea\_rt.jar=61017:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.3\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath C:\Users\Roystan\IdeaProjects\JavaWorkspace\out\production\JavaWorkspace Collections.Queue.ArrayDequeClass

Animals: [Cat, Cat, Giraffe, Dog, Horse, Giraffe, Dog, Rhino, Giraffe, Giraffe]offerValue trueaddValuetrue

Animals after firstElement [Cat, Cat, Giraffe, Dog, Horse, Giraffe, Dog, Rhino, Giraffe, Giraffe]firstElement Cat

Animals after LastElement [Cat, Cat, Giraffe, Dog, Horse, Giraffe, Dog, Rhino, Giraffe, Giraffe]LastElement Giraffe

Animals after elementPeek [Cat, Cat, Giraffe, Dog, Horse, Giraffe, Dog, Rhino, Giraffe, Giraffe]elementPeek Cat

Animals after firstElementPeek [Cat, Cat, Giraffe, Dog, Horse, Giraffe, Dog, Rhino, Giraffe, Giraffe]firstElementPeek Cat

Animals after lastElementPeek [Cat, Cat, Giraffe, Dog, Horse, Giraffe, Dog, Rhino, Giraffe, Giraffe]lastElementPeek Giraffe

Animals after elementRemove [Cat, Giraffe, Dog, Horse, Giraffe, Dog, Rhino, Giraffe, Giraffe]elementRemove Cat

Animals after firstElementRemove [Giraffe, Dog, Horse, Giraffe, Dog, Rhino, Giraffe, Giraffe]firstElementRemove Cat

Animals after lastElementRemove [Giraffe, Dog, Horse, Giraffe, Dog, Rhino, Giraffe]lastElementRemove Giraffe

Animals after elementpop [Dog, Horse, Giraffe, Dog, Rhino, Giraffe]elementpop Giraffe

Animals after push [Tiger, Dog, Horse, Giraffe, Dog, Rhino, Giraffe]

Animals after Elementpoll [Dog, Horse, Giraffe, Dog, Rhino, Giraffe]Elementpoll Tiger

Animals after firstElementpoll [Horse, Giraffe, Dog, Rhino, Giraffe]firstElementpoll Dog

Animals after lastElementpoll [Horse, Giraffe, Dog, Rhino]lastElementpoll Giraffe

Animals after element [Horse, Giraffe, Dog, Rhino]element Horse

Animals after firstOcurenceElement [Horse, Dog, Rhino]firstOcurenceElement true

Animals after lastOcurenceElement [Horse, Dog, Rhino]lastOcurenceElement false

Animals after DescendingIterator [Horse, Dog]

Animals after animalClone [Horse, Dog, Rhino]

hasAnimalEquals true

hasAnimalEquals true

hashCode() 2046562095

widerLivingCreatures [Snake, fishes, Dinoaueraus]removeif true

hasAnimalContains true

Animals after AddAll [Horse, Dog, Rhino, Horse, Dog, Rhino, Snake, fishes, Dinoaueraus]

Animals after retainAll [Snake, fishes, Dinoaueraus]

Animals after removeAll []

Animals after isEmpty true

Animals after New Elemets added [reptiles, amphibions, birds, fishes, Other creatures]

Animals after AscendingIterator [reptiles, amphibions, birds, fishes, Other creatures]

reptiles

amphibions

birds

fishes

Other creatures

ArrayValues

index 0Arrayvalues reptiles

index 1 Arrayvalues amphibions

index 2 Arrayvalues birds

index 3 Arrayvalues fishes

index 4 Arrayvalues Other creatures

Iterated values

reptiles amphibions birds fishes Other creatures

descending values

Other creatures birds amphibions reptiles

Animals [reptiles, birds, Other creatures]

Animals after clear []

Process finished with exit code 0