Systembeskrivning

länkList

node first & last: Definierar den första och sista noden.

Size int: Definierar storleks heltal.

class node: Definierar vad en node är genom att tilldela ett värde ”String” och en pekare ”next”.

Removefirst: Tar bort den första noden från listan genom att göra den andra noden i listan till den första. Size blir mindre . Om listan är tom så sätter den first och last till null.

RemoveLast: Tar bort den sista noden genom att gå igenom alla noder tills den påträffar en node som pekar på den sista noden. Sedan så ändrar de den näst sista nodens pekare till null och på så sätt gör den till sist. Size blir en mindre. Om listan är tom så tilldelar den first och last värdet null.

Addfirst: Lägger till en node längst fram i listan. Skapar en node och lägger till en till size. Om listan är tom så läggs den nya noden längst fram i listan. Om den inte är tom så gör man att den nya noden pekar mot den förra first och sätter den nya noden till first.

Isempty: Om first är null så returnerar den true.

Addlast: Skapar en ny node och tilldelar den ett värde. Pekaren för denna node ändras till null och storlek ökas med en. Om listan är tom så tilldelas first och last den nya noden.

(Test metod)printString: Går igenom hela listan och lägger till de i en string. Detta retuneras sedan.

nodeToString: Tar en node, tar ut dess string data och returnerar det.

Getize: Retunerar storleken på listan. Om den är mindre än **0** så sätter den tillbaka size till noll och skriver ut i terminalen att det blev fel någonstans.

Skrivarkö:

Länkar till länkList klassen så man kan använda den i skrivarkön genom (i.).

Enqueue: Tar in en string och lägger till den i listan via addlast.

isEmpty: Returnerar true om den är tom via isEmpty.

Dequeue: Skapar en temp String som lagrar den förssta nodens värde. Den första noden tas bort genom **removeFirst** metoden och sedan retuneras det lagrade värdet.

Size: Retunerar Size genom **getSize** metoden.

Testkörningar

```
skrivarKö kööö = new skrivarKö();

kööö.enqueue("damn1");
kööö.enqueue("damn2");
kööö.enqueue("damn3");

//dequeues & prints. Prints out the size after the dequeue
for (int i = kööö.size(); i > 0; i--){
   print( "Skrivs ut:    " + kööö.dequeue());
   print("How many Left: " + Integer.toString(kööö.size()));
}
```

Länkar kön, lägger till 3 olika nodes i kön. Skriver ut hur många nodes det finns i kön.

En for loop som körs så länge i är större än noll. I är storleken på kön.

Den skriver ut den som tas bort och hur många som är kvar i kön. Efter varje gången den kört så tas en bort från i.

Terminalen:

```
Skrivs ut:    damn1
How many Left: 2
Skrivs ut:    damn2
How many Left: 1
Skrivs ut:    damn3
How many Left: 0
```

```
if (köö.isEmpty() == true) {
    print("The list is empty list");
} else {
    print("List is not empty");
}
```

Om kön är tom så skriver den ut att listan är tom annars så skriver den ut att listen inte är tom.

Terminal:

```
List is not empty
```

```
The list is empty list
```

Och om man försöker ta bort från en tom lista så retuneras inget.

```
köö.dequeue();
```

Terminalen:

```

```