**QueueDrvr Class**

**QueueDrvr()**

Initialize 3 new queues called demQueue, repQueue, and whigQueue using ArrayList and a StackDrvr object

//end constructor

**isEmptyDem()**

return true if demElem is 0

//end isEmptyDem()

**isEmptyRep()**

return true if repElem is 0

//end isEmptyRep

**isEmptyWhig()**

return true if whigElem is 0

//end isEmptyWhig

**insert (Presidents president)**

Swtich(party)

Case: “Democrat”

Call insertDem(president)

Case: “Republican”

Call insertRep(president)

Case: “Whig”

Call insertWhig(president)

//end switch

//end insert

**insertDem(Presidents president)**

set next item in demQueue to president passed as parameter

increment rear

increment elements

//end insertDem

**insertRep(Presidents president)**

set next item in repQueue to president passed as parameter

increment rear

increment elements

//end insertRep

**insertWhig(Presidents president)**

set next item in whigQueue to president passed as parameter

increment rear

increment elements

//end insertWhig

**display(int dispType)**

switch (dispType)

case: unsorted

call getQHeader(dem)

call displayDem()

call getQHeader(rep)

call displayRep()

call getQHeader(whig)

call displayWhig()

case: sorted

call getPQHeader(dem)

call displayDem()

call getPQHeader(rep)

call displayRep()

call getPQHeader(whig)

call displayWhig()

case: updated

call getPQHeader(demUpdate)

displayDem()

call getPQHeader(repUpdate)

displayRep()

call get PQHeader(whigUpdate)

displayWhig()

//end switch

//end display()

**getQHeader(int type)**

switch(type)

case 1: Democrat

print report title

case2: Republican

print report title

case3: Whig

print report title

//end switch

Print formatted header

//end getQHeader

**getPQHeader (int type)**

switch(type)

case 1: Democrat

print report title

case2: Republican

print report title

case3: Whig

print report title

case 4: updated Democrat

print report title

case 5: updated Republican

print repot title

case 6: updated Whig

print report title

//end switch

Print formatted header

//end getPQHeader

**displayDem()**

for (each index in demQueue, set counter to first index, increment counter)

print Presidents object in each index of demQueue to a formatted string

//end for

//end displayDem()

**displayRep()**

for (each index in repQueue, set counter to first index, increment counter)

print Presidents object in each index of repQueue to a formatted string

//end for

//end displayRep()

**displayWhig()**

for (each index in whigQueue, set counter to first index, increment counter)

print Presidents object in each index of whigQueue to a formatted string

//end for

//end displayWhig()

**selectionSort()**

call sortDem()

call sortRep()

call sortWhig()

//end selection sort

**sortDem()**

declare out, counter for outer loop

declare in, counter for inner loop

declare min, the smallest value in queue

for (out = 0, while out less than demElem-1, increment out)

set min to out

for(in = out+1, while in less than demElem, increment in)

if (President object number at position in is less than president object number at position out

set min to in

//end if

//end for

swapDem(out, min)

//end for

//end sortDem

**sortRep()**

declare out, counter for outer loop

declare in, counter for inner loop

declare min, the smallest value in queue

for (out = 0, while out less than repElem-1, increment out)

set min to out

for(in = out+1, while in less than repElem, increment in)

if (President object number at position in is less than president object number at position out

set min to in

//end if

//end for

swapRep(out, min)

//end for

//end sortRep

**sortWhig()**

declare out, counter for outer loop

declare in, counter for inner loop

declare min, the smallest value in queue

for (out = 0, while out less than whigElem-1, increment out)

set min to out

for(in = out+1, while in less than whigElem, increment in)

if (President object number at position in less than president object number at position out

set min to in

//end if

//end for

swapWhig(out, min)

//end for

//end sortWhig

**swapDem(int one, int two)**

declare Presidents object named temp and set to position one in demQueue

set president object in position one to position two

set president object in position two to temp

//end swapDem()

**swapRep(int one, int two)**

declare Presidents object named temp and set to position one in repQueue

set president object in position one to position two

set president object in position two to temp

//end swapDem()

**swapWhig(int one, int two)**

declare Presidents object named temp and set to position one in whigQueue

set president object in position one to position two

set president object in position two to temp

//end swapWhig()

**getUpdate(Presidents president)**

push the president passed as parameter onto the update stack

//end get update

**update()**

for (each index in update stack, set counter to first index, increment counter)

pop a president off the stack

switch(popPrez.getParty())

case: “Democrat”

call updateDem(popPrez)

case: “Republican”

call updateRep(popPrez)

case: “Whig”

call updateWhig(popPrez)

//end swtich

//end for

//end update()

**updateDem(President popPrez)**

declare counter var i

if(demQue is empty)

add the popPrez to demQueue

else

for (i = demElem -1, i is greater or equal to 0, decrement i)

if (popPrez number is less than or equal to president number at index i of demQueue)

set president at index i to index i+1

remove president at index i

else

break loop

//end if

//end for

Set popPrez to index i+1 of demQueue

Increment demRear

Increment demElem

//end if

//end updateDem

**updateRep(President popPrez)**

declare counter var k

if(repQue is empty)

add the popPrez to repQueue

else

for (k = repElem -1, k is greater or equal to 0, decrement k)

if (popPrez number is less than or equal to president number at index k of repQueue)

set president at index k to index k+1

remove president at index k

else

break loop

//end if

//end for

Set popPrez to index k+1 of repQueue

Increment repRear

Increment repElem

//end if

//end updateRep

**updateWhig(President popPrez)**

declare counter var l

if(whigQue is empty)

add the popPrez to whigQueue

else

for (l = whigElem -1, l is greater or equal to 0, decrement l)

if (popPrez number is less than or equal to president number at index lof whigQueue)

set president at index k to index k+1

remove president at index k

else

break loop

//end if

//end for

Set popPrez to index l+1 of whigQueue

Increment whigRear

Increment whigElem

//end if

//end updateWhig

**displayStack()**

display update Stack

//end display

//end QueueDrvr