**Bully.py**

from statistics import mode

class Process:

  def \_\_init\_\_(self, process\_id, total\_count):

    self.process\_id = process\_id

    self.total\_count = total\_count

    self.leader\_id = -1

    self.is\_active = True

  def crash(self):

    self.is\_active = False

  def start(self):

    self.is\_active = True

  def is\_leader(self):

    if self.process\_id == self.leader\_id:

      return True

    return False

  def set\_leader(self, leader):

    self.leader\_id = leader

  def get\_leader(self):

    return self.leader\_id

  def sendRequest(self, toProcess):

      print(f"Sending request to process {toProcess.process\_id} from {self.process\_id}")

      if(toProcess.reciveRequest(self.process\_id)):

        print(f"Ok recived from {toProcess.process\_id}")

        self.set\_leader(toProcess.process\_id)

      else:

        print(f"No response from {toProcess.process\_id}")

  def reciveRequest(self, fromProcess):

    if(self.is\_active):

      print(f"Recived request from process {fromProcess}.")

      return self.recivedMessage()

    return False

  def recivedMessage(self):

    return True;

class Bully:

    def \_\_init\_\_(self, total\_count):

      self.processes = []

      self.total\_count = total\_count

  # self.leader = None

    def intiailzeProcesses(self):

      self.processes = []

      for i in range(self.total\_count):

        self.processes.append(Process(i, total\_count = self.total\_count))

      self.elect\_leader()

      self.coordinator()

    def elect\_leader(self, current=0):

      for i in range(current, self.total\_count):

        if self.processes[i].is\_active:

          for j in range(i+1, self.total\_count):

            if(self.processes[j].is\_active):

              self.processes[i].sendRequest(self.processes[j])

            elif(not self.processes[j].is\_active and i+1==self.total\_count-1):

              self.processes[i].sendRequest(self.processes[i])

      if self.processes[i].get\_leader()==-1:

        self.processes[i].sendRequest(self.processes[i])

  # if(i==self.total\_count-1):

  # self.processes[i].sendRequest(self.processes[i])

    def crash(self, crash\_id):

      if(crash\_id<self.total\_count and crash\_id>=0):

        self.processes[crash\_id].crash()

  # print(f"Process id {Process.process\_id} crashed.")

        if(self.processes[crash\_id].is\_leader()):

          print("Leader process Down.\n Initaling the leader lookout.")

          self.elect\_leader(0)

    def start(self, process\_id):

      if(self.processes[process\_id].is\_active):

        print("Process already active")

      else:

        self.processes[process\_id].start()

        self.elect\_leader()

  # if(self.processes[process\_id].is\_active):

  # if process\_id>self.processes[self.leader].get\_leader():

  # self.elect\_leader(self.leader)

    def coordinator(self):

      leader = []

      for p in self.processes:

        if p.is\_active:

          print(p.get\_leader())

          leader.append(p.get\_leader())

      self.leader = mode(leader)

**Driver.py**

from statistics import mode

class Process:

  def \_\_init\_\_(self, process\_id, total\_count):

    self.process\_id = process\_id

    self.total\_count = total\_count

    self.leader\_id = -1

    self.is\_active = True

  def crash(self):

    self.is\_active = False

  def start(self):

    self.is\_active = True

  def is\_leader(self):

    if self.process\_id == self.leader\_id:

      return True

    return False

  def set\_leader(self, leader):

    self.leader\_id = leader

  def get\_leader(self):

    return self.leader\_id

  def sendRequest(self, toProcess):

      print(f"Sending request to process {toProcess.process\_id} from {self.process\_id}")

      if(toProcess.reciveRequest(self.process\_id)):

        print(f"Ok recived from {toProcess.process\_id}")

        self.set\_leader(toProcess.process\_id)

      else:

        print(f"No response from {toProcess.process\_id}")

  def reciveRequest(self, fromProcess):

    if(self.is\_active):

      print(f"Recived request from process {fromProcess}.")

      return self.recivedMessage()

    return False

  def recivedMessage(self):

    return True;

class Bully:

    def \_\_init\_\_(self, total\_count):

      self.processes = []

      self.total\_count = total\_count

  # self.leader = None

    def intiailzeProcesses(self):

      self.processes = []

      for i in range(self.total\_count):

        self.processes.append(Process(i, total\_count = self.total\_count))

      self.elect\_leader()

      self.coordinator()

    def elect\_leader(self, current=0):

      for i in range(current, self.total\_count):

        if self.processes[i].is\_active:

          for j in range(i+1, self.total\_count):

            if(self.processes[j].is\_active):

              self.processes[i].sendRequest(self.processes[j])

            elif(not self.processes[j].is\_active and i+1==self.total\_count-1):

              self.processes[i].sendRequest(self.processes[i])

      if self.processes[i].get\_leader()==-1:

        self.processes[i].sendRequest(self.processes[i])

  # if(i==self.total\_count-1):

  # self.processes[i].sendRequest(self.processes[i])

    def crash(self, crash\_id):

      if(crash\_id<self.total\_count and crash\_id>=0):

        self.processes[crash\_id].crash()

  # print(f"Process id {Process.process\_id} crashed.")

        if(self.processes[crash\_id].is\_leader()):

          print("Leader process Down.\n Initaling the leader lookout.")

          self.elect\_leader(0)

    def start(self, process\_id):

      if(self.processes[process\_id].is\_active):

        print("Process already active")

      else:

        self.processes[process\_id].start()

        self.elect\_leader()

  # if(self.processes[process\_id].is\_active):

  # if process\_id>self.processes[self.leader].get\_leader():

  # self.elect\_leader(self.leader)

    def coordinator(self):

      leader = []

      for p in self.processes:

        if p.is\_active:

          print(p.get\_leader())

          leader.append(p.get\_leader())

      self.leader = mode(leader)

**Ring.py**

class Pro:

  def \_\_init\_\_(self, id):

    self.id = id

    self.act = True

class GFG:

  def \_\_init\_\_(self):

    self.TotalProcess = 0

    self.process = []

  def initialiseGFG(self):

    print("No of processes 5")

    self.TotalProcess = 5

    self.process = [Pro(i) for i in range(self.TotalProcess)]

  def Election(self):

    print("Process no " + str(self.process[self.FetchMaximum()].id) + " fails")

    self.process[self.FetchMaximum()].act = False

    print("Election Initiated by 2")

    initializedProcess = 2

    old = initializedProcess

    newer = old + 1

    while (True):

      if (self.process[newer].act):

        print("Process " + str(self.process[old].id) + " pass Election(" + str(self.process[old].id) + ") to" + str(self.process[newer].id))

        old = newer

      newer = (newer + 1) % self.TotalProcess

      if (newer == initializedProcess):

        break

    print("Process " + str(self.process[self.FetchMaximum()].id) + " becomes coordinator")

    coord = self.process[self.FetchMaximum()].id

    old = coord

    newer = (old + 1) % self.TotalProcess

    while (True):

      if (self.process[newer].act):

        print("Process " + str(self.process[old].id) + " pass Coordinator(" + str(coord) + ") message to process " + str(self.process[newer].id))

        old = newer

      newer = (newer + 1) % self.TotalProcess

      if (newer == coord):

        print("End Of Election ")

        break

  def FetchMaximum(self):

    maxId = -9999

    ind = 0

    for i in range(self.TotalProcess):

      if (self.process[i].act and self.process[i].id > maxId):

        maxId = self.process[i].id

        ind = i

    return ind

def main():

  object = GFG()

  object.initialiseGFG()

  object.Election()

if \_\_name\_\_ == "\_\_main\_\_":

  main()